175 PTAS
57
mi computer
CURSO PRACTICO DEL ORDENADOR PERSONAL,
EL MICRO Y EL MINIORDENADOR
Editorial Delta, S.A.

MI COMPUTER CURSO PRACTICO
DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona

Volumen V - Fascículo 57

Director:	José Mas Godayol
Director editorial:	Gerardo Romero
Jefe de redacción:	Pablo Parra
Coordinación editorial:	Jaime Mardones Francisco Martín
Asesor técnico:	Ramón Cervelló

Redactores y colaboradores: G. Jefferson, R. Ford, F. Martín, S. Tarditti, A. Cuevas, F. Blasco

Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Paseo de Gracia, 88, 5.°, 08008 Barcelona
Tels. (93) 215 10 32 / (93) 215 10 50 - Télex 97848 EDLTE

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador*, se publica en forma de 96 fascículos de aparición semanal, encuadernables en ocho volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-7598-007-4 (tomo 5)
84-85822-82-X (obra completa)
Depósito Legal: B. 52/1984

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 138502
Impreso en España - Printed in Spain - Febrero 1985

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, 28034 Madrid.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Edificio Bloque Dearmas, final Avda. San Martín con final Avda. La Paz, Caracas 1010.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (96 fascículos más las tapas, guardas y transferibles para la confección de los 8 volúmenes) son las siguientes:

a) Un pago único anticipado de 19 425 ptas. o bien 8 pagos trimestrales anticipados y consecutivos de 2 429 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 6.850.277 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Editorial Delta, S.A. (Paseo de Gracia, 88, 5.°, 08008 Barcelona), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Editorial Delta, S.A., en la forma establecida en el apartado b).

Para cualquier aclaración, telefonear al (93) 215 75 21.

No se efectúan envíos contra reembolso.





Por la senda correcta

Una vez visto cómo se puede mover un robot, vamos a analizar las formas de controlar su desplazamiento



De laberintos y ratones

Ratón... Las competiciones de Micro Mouse, en las cuales compiten ratones robots en el recorrido de un laberinto, han sido una valiosa fuente de conocimiento práctico y experiencia técnica para muchos aficionados a los robots. El Quester de David Buckley, que vemos en la fotografía, lleva una amplia gama de sensores (ópticos, sonoros y sensibles al tacto)

Y laberinto Cada microrratón tiene un período de práctica en el cual "aprende" el trazado del laberinto por cualquier método que no requiera comunicación externa, y luego debe recorrerlo contra reloj, siendo el objetivo básico llegar al centro del laberinto en el menor tiempo posible

Marcus Willy

El procedimiento más sencillo para desplazar un robot implica la utilización de un dispositivo mecánico que "lee" una ficha de forma especial insertada en aquél. El contorno de la ficha es seguido por un pequeño detector que a su vez opera una serie de palancas para controlar la dirección del ingenio electrónico. En el pasado se podían adquirir coches en miniatura y pequeños robots de juguete que funcionaban de esta forma. La ficha que contenía el programa se creaba utilizando unas tijeras para cortar. El autómata se movía de acuerdo al contorno del borde recortado.

Otros robots empleaban dispositivos que les permitían seguir un recorrido establecido mediante relés electromecánicos internos. Sin embargo, la aplicación de estos procedimientos mecánicos para el control del movimiento era limitada, por la sencilla razón de que los componentes mecánicos suelen ser caros y relativamente inexactos. Pero sí proporcionan un precedente para los métodos actuales.

Uno de los mejores procedimientos empleados hoy en día se basa en hacer que el robot siga una pista especialmente trazada para él en el suelo. Esto es similar al método que utilizan los juegos de coches de carreras, que poseen una patilla guía insertada en una ranura continua de una pista de carreras a escala. Las dos formas más corrientes de autómatas cibernéticos que siguen una pista son, no obstante, aquellos que se desplazan a través de una línea dibujada en el suelo y los que se guían por medio de un cable.

En los robots que siguen una línea se utiliza un sensor de luz, normalmente una célula fotoeléctrica o un sensor infrarrojo, para determinar si el robot está situado sobre una zona clara o una zona oscura. Si el color del suelo es oscuro y el de la línea es claro, la salida del sensor siempre estará en su nivel más alto cuando el sensor esté directamente sobre la línea. Por consiguiente, si el robot sigue siempre la ruta que proporciona la salida eléctrica más elevada del sensor, en ningún momento se apartará de la línea trazada.

Esta técnica plantea un problema: ¿qué hace el robot cuando la salida del sensor cae, indicando que ha abandonado la línea? Con un sistema de un único sensor, lo mejor que puede hacer el robot es dar vueltas en derredor hasta que la señal de salida del sensor vuelva a subir, indicando que se halla nuevamente sobre la línea. Luego puede continuar en la dirección en la que esté encarado. Este sistema no es tan aleatorio como podría parecer. Por ejemplo, si el robot estuviera yendo hacia la iz-

Codificador



Un codificador de eje permite que un robot sepa cuánto han rotado los ejes de sus ruedas. El dispositivo es un disco calibrado que se coloca en un eje. El plato circular está dividido en cierto número de círculos concéntricos, marcados cada uno en zonas que son opacas o transparentes. Con una fuente luminosa y un sensor de luz para cada anillo se puede calcular la orientación exacta del eje.
La precisión de un codificador de eje depende del número de anillos de que disponga el disco. En nuestra ilustración vemos un codificador de eje de tres anillos, que puede codificar los números binarios del 000 al 111 (decimales del 0 al 7). La precisión de este codificador es de 360/8=45 grados. Ocho anillos concéntricos darían una precisión de 1,41 grados

quierda cuando la salida del sensor cayera, lo lógico entonces es que gire hacia la derecha para volver a encontrar la línea. Asimismo, habiendo hallado ésta es lógico que suponga que la dirección en la cual debe dirigirse ahora esté en algún punto entre el recorrido que estaba siguiendo cuando perdió la línea (izquierda) y el recorrido que ha debido realizar para hallarla otra vez (derecha).

Un sistema que reduce el tiempo que tiene que perder un robot "extraviado" tratando de hallar otra vez la dirección utiliza dos sensores orientados a cada uno de los lados de la línea. Esto significa que cuando el robot está sobre la línea, la salida de ambos sensores es baja. Si el robot comienza a apartarse de ésta, la salida de uno de los sensores se eleva. Ello significa que el robot sabe inmediatamente que se ha desviado y en qué dirección ha cometido el error. Si se apartara hacia la derecha, se elevaría la salida del sensor izquierdo y tomaría esto como una señal para girar hacia la izquierda, lo que lo devolvería nuevamente a su recorrido.

Este sistema no requiere tener una línea blanca sobre un fondo negro, sino que puede funcionar igualmente bien con una línea oscura sobre un fondo claro. Lo importante es el contraste y que la programación le diga al robot qué hacer cuando un sensor lee un valor incorrecto.

El otro sistema que se utiliza para los robots que siguen huellas implica enviar una pequeña corriente eléctrica a lo largo de un cable colocado sobre el suelo. Esta corriente genera un pequeño campo magnético alrededor del cable, que es detectado por el sensor. Éste no necesita ser un sensor complicado: una pequeña bobina de alambre captará el campo magnético y producirá un pequeño voltaje que se puede después ampliar y que actuará exactamente de la misma manera en que lo hacen los sensores. Con frecuencia, los robots industriales que necesitan desplazarse se basan en un cable enterrado en el suelo debajo de ellos. Si dependieran de una línea pintada sobre la superficie, todo iría bien hasta que el suelo se ensuciara.

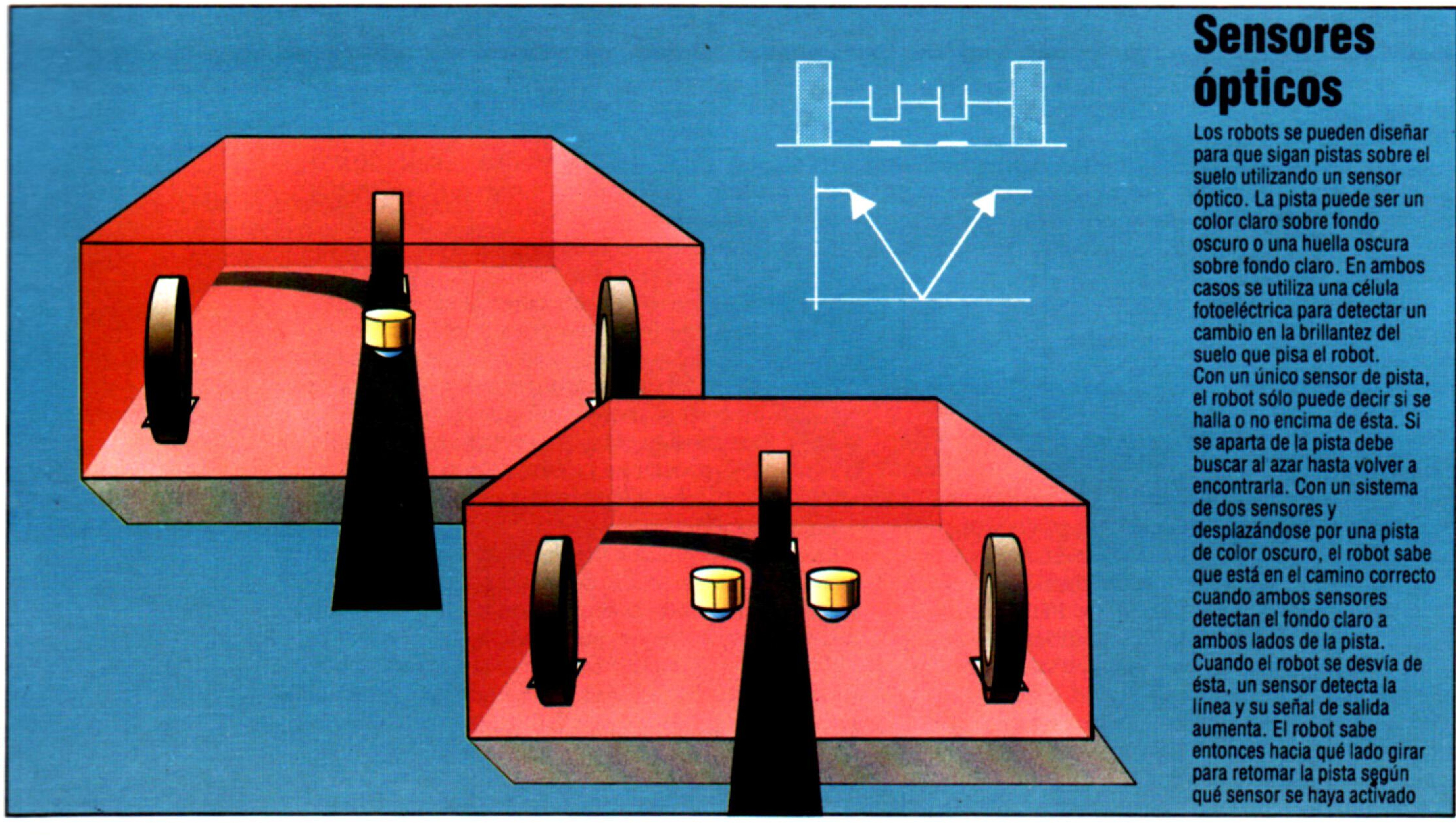
Control remoto

Otro método es el de que un operador humano controle al robot a distancia. Esto es particularmente útil cuando las tareas a llevar a cabo por el robot podría realizarlas igualmente un ser humano, pero en un entorno demasiado hostil y poco seguro. Ejemplos de ello son la desactivación de bombas, la manipulación de materiales radiactivos o materiales químicos peligrosos y el trabajo en zonas demasiado calientes, frías o peligrosas para un ser humano.

Un robot de esta clase muy conocido es el soviético Lunojod I, depositado en la superficie lunar por el Luma 16 en 1970. Se trataba de un robot sobre ruedas que recogió información sobre la superficie del satélite bajo el control por radio de científicos situados en la Tierra.

El control de robots de este tipo es algo diferente del que se ejerce sobre un avión a escala por medio de la radio. La señal emitida por ésta puede ser analógica, cuya intensidad varía a tenor del movimiento que se requiera del robot, o digital, que compone un patrón de bits que ofrece detalles relativos a los movimientos a efectuar. Las comunicaciones analógicas tienden a ser menos eficaces que los métodos digitales, porque hay diversos factores que podrían interferir en la intensidad de la señal de una transmisión analógica. Pruebe a escuchar una emisora de radio lejana y observe cómo varía la recepción según la hora del día y las condiciones climatológicas. El mismo tipo de problemas puede afectar a las señales utilizadas para comunicaciones con robots.

Los métodos digitales también pueden plantear



Sensores ópticos

Los robots se pueden diseñar para que sigan pistas sobre el suelo utilizando un sensor óptico. La pista puede ser un color claro sobre fondo oscuro o una huella oscura sobre fondo claro. En ambos casos se utiliza una célula fotoeléctrica para detectar un cambio en la brillantez del suelo que pisa el robot.
Con un único sensor de pista, el robot sólo puede decir si se halla o no encima de ésta. Si se aparta de la pista debe buscar al azar hasta volver a encontrarla. Con un sistema de dos sensores y desplazándose por una pista de color oscuro, el robot sabe que está en el camino correcto cuando ambos sensores detectan el fondo claro a ambos lados de la pista. Cuando el robot se desvía de ésta, un sensor detecta la línea y su señal de salida aumenta. El robot sabe entonces hacia qué lado girar para retomar la pista según qué sensor se haya activado

Kevin Jones



problemas, en especial cuando la interferencia hace que se pierdan bits o que se los inserte incorrectamente. Para evitar esto, los mensajes a los robots se suelen repetir varias veces. Estos autómatas cibernéticos sólo se ponen en acción luego de haber recibido reiteradamente un mensaje enviado en idénticos términos.

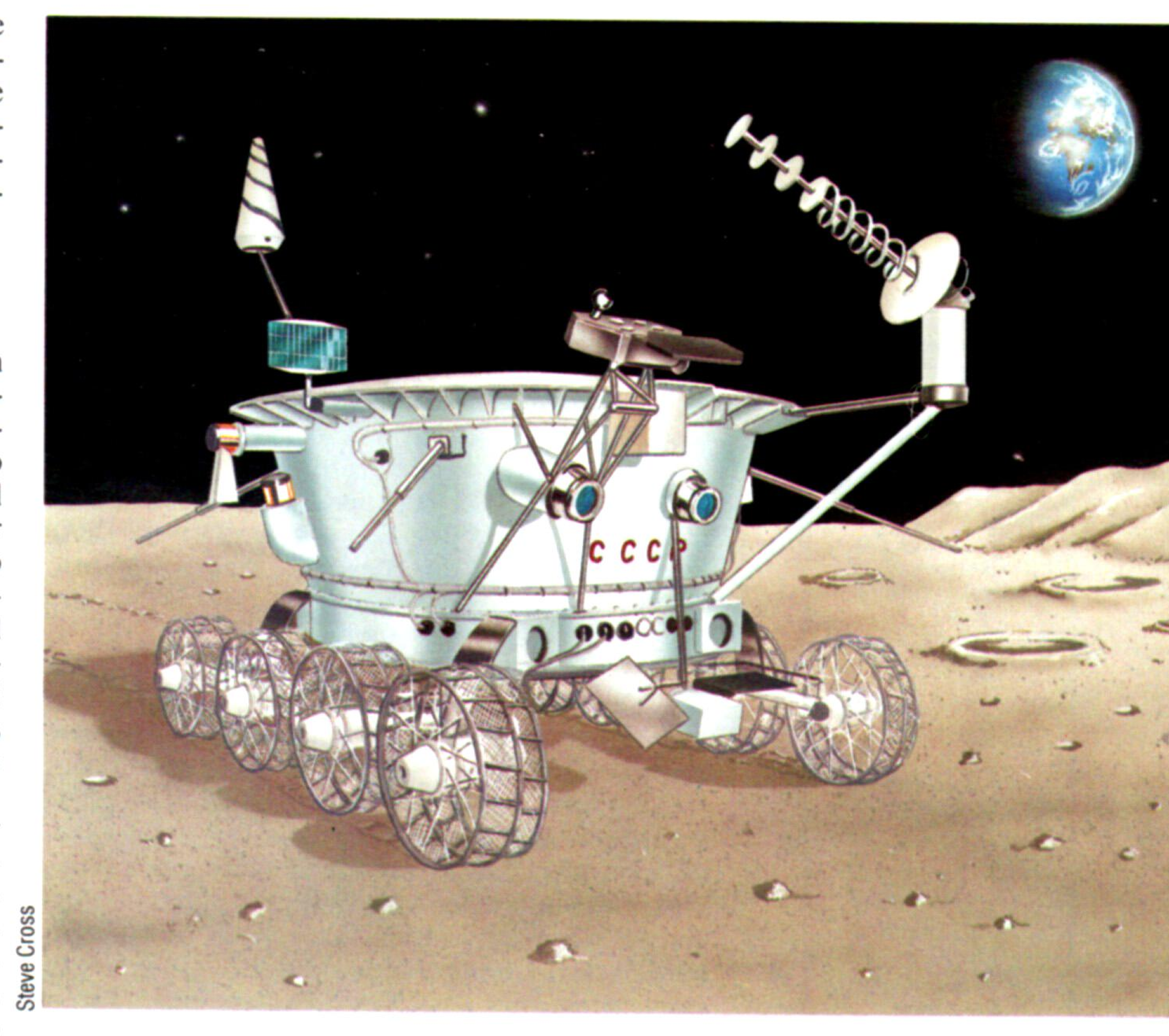
Sistemas de realimentación

Una técnica más sofisticada es la utilización de un sistema de "bucle", en el cual el robot le proporciona al transmisor un feedback (retorno, realimentación) relativo a la señal que acaba de recibir. Esto se podría considerar como un diálogo entre el transmisor y el robot. Por ejemplo, el transmisor puede decir "hacia adelante" y, habiendo recibido el mensaje, el robot le dirá "¿has dicho hacia adelante?", a lo que el transmisor respondería "sí", y el robot se movería entonces de acuerdo a la indicación. Esto podría ayudar a evitar errores graves si el robot estuviera manipulando residuos nucleares o estuviera a punto de caerse dentro de un cráter de la Luna.

Las mismas técnicas generales se pueden aplicar a otros medios de control remoto. Por ejemplo, algunos robots se pueden controlar a través de emisores infrarrojos de la clase que se utiliza para los dispositivos de control a distancia para aparatos de televisión. O se podrían controlar mediante sonido ultrasónico, algo así como un silbato para perros, o mediante sonido audible de una naturaleza distintiva, tales como una serie de palmadas. Sea cual sea el método que se utilice, las técnicas que se aplican para transmitir el mensaje y asegurar que el robot lo haya recibido son las mismas.

Si el operador humano se hallara bastante cerca del robot, tal vez no fuera necesario emplear técnicas tan sofisticadas: al robot se le podrían transmitir las órdenes a través de un cable de conexión. También existe la posibilidad de utilizar más de un cable, lo que equivaldría a tener varios canales en el caso de un avión controlado por radio. Pero, en el caso del robot, los cables adicionales generalmente se emplean para proporcionar una comunicación en paralelo en vez de en serie (se envía una serie de bits en paralelo a lo largo de los cables en vez de enviarse como una serie de impulsos a lo largo de un cable). Esto permite una comunicación más rápida con el robot. Quizás aún más importante sea el hecho de que la mayoría de los ordenadores disponen de una puerta en paralelo. Ésta comunica de forma excelente las instrucciones al robot desde el teclado de un ordenador.

Si el movimiento del robot lo ha de controlar un operador humano sentado frente al teclado de un ordenador y puede ver al robot, entonces en principio hay muy poca diferencia entre controlar al robot a través de un operador humano y hacerlo por medio de un ordenador. Ello se debe a que, al igual que el avión controlado por radio, el operador siempre puede ver lo que está haciendo el robot y corregir de inmediato cualquier error. Pero si el autómata cibernético se halla a cierta distancia (en la Luna, p. ej., e incluso en la habitación contigua), o si se ha de controlar a través de un programa dentro del ordenador en vez de a través de instrucciones de teclado en tiempo real, entonces el robot ha de ser ligeramente más inteligente.



Steve Cross

Un salto de gigante para los robots
El Lunojod I fue depositado en la superficie lunar por la URSS en 1970 para recoger información acerca de la naturaleza de la superficie y la atmósfera del satélite terrestre. No era un auténtico robot (se lo controlaba por radio desde la Tierra), pero su disposición invulnerable e invariable frente a las rigurosas condiciones lunares permitió que la nave espacial que lo llevó pudiera regresar con una aportación científica superior de la que hubiera sido posible con astronautas y sus imprescindibles y elaborados sistemas de acondicionamiento. Al igual que todos los objetos del espacio controlados a distancia, el Lunojod experimentaba un intervalo de demora de tres segundos entre su transmisión de información a la Tierra y la recepción de una señal de control en respuesta

Esencialmente, lo que se necesita es alguna forma de realimentación (*feedback*). Esto es un proceso que permite que el sistema ajuste lo que está haciendo en relación a lo que ya ha efectuado y a lo que debería estar realizando. Por ejemplo, si desea que un robot recorra tres metros por una habitación y lo está controlando directamente, puede empezar a moverlo, evaluar su progreso y detenerlo cuando lleve recorridos tres metros. Esto es así porque usted tiene una realimentación visual con el robot: ve hasta dónde ha avanzado, hasta qué distancia desea que vaya, y puede corregir sus acciones en consecuencia.

En ausencia de la realimentación sensorial humana, el robot ha de disponer de alguna propia si es que debe desplazarse con precisión. El robot que sigue una línea utiliza la realimentación de la línea que está siguiendo en el suelo y, del mismo modo, el robot controlado por ordenador debe emplear alguna realimentación si es que ha de recorrer exactamente tres metros hacia adelante. Uno de los métodos más comúnmente aplicados para proporcionar la realimentación necesaria es un *codificador de eje*, que consiste en un disco circular fijado a los ejes principales de las ruedas del robot y que proporciona una medida muy exacta de cuánto han rotado. Por lo tanto, si el ordenador le envía instrucciones al robot indicándole que avance tres metros, el robot puede empezar a moverse y, al mismo tiempo, controlar las señales provenientes de sus codificadores de eje para ver hasta dónde se ha desplazado. Si el robot debe avanzar más, puede continuar desplazándose. Cuando llegue allí se puede detener, y si por cualquier motivo no acaba en el lugar correcto, entonces siempre se puede apoyar en la cantidad correcta calculada a partir de la información enviada por los codificadores de eje para corregir su error.

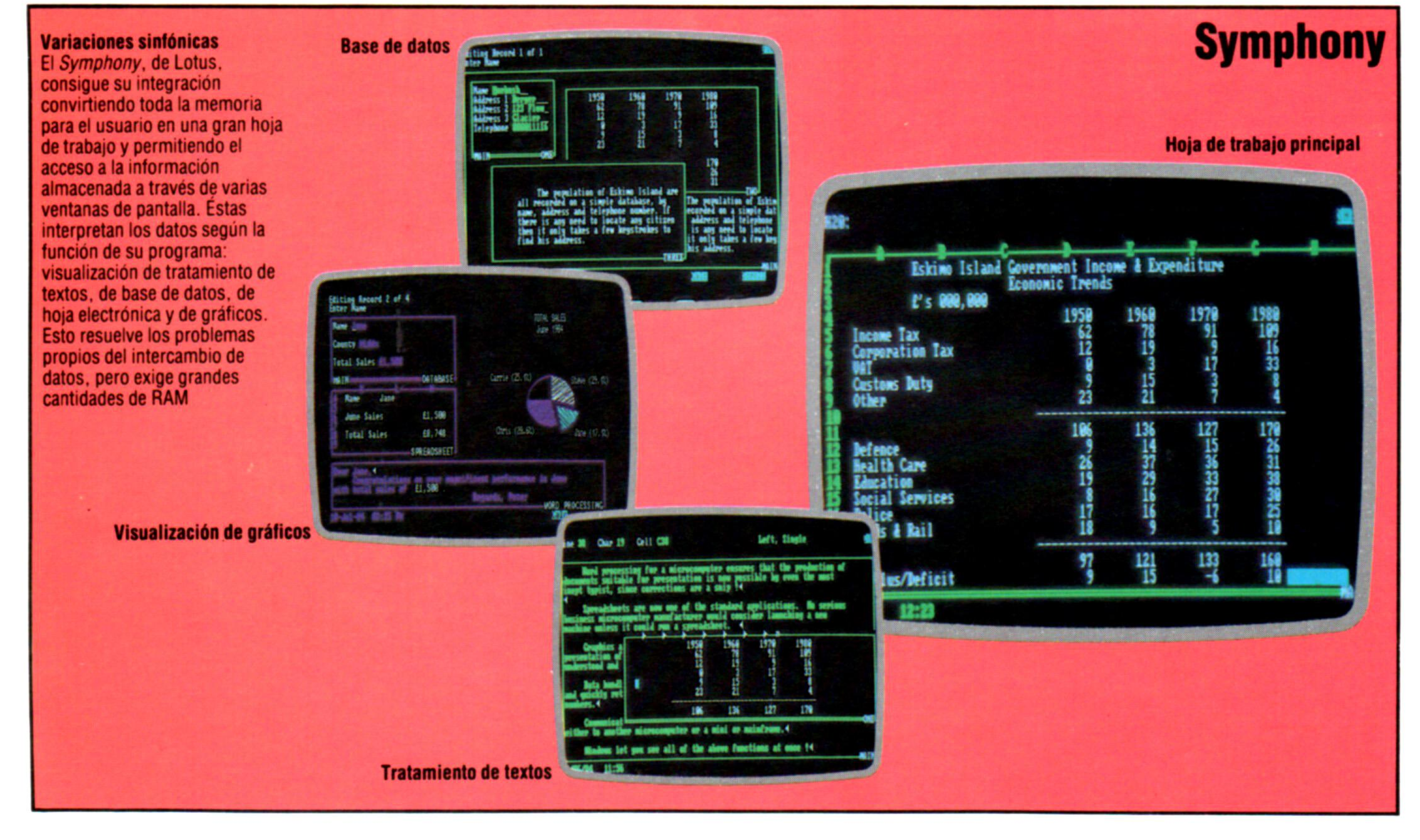
Sinfonía en software

Examinemos tres paquetes de software integrado, cuyas técnicas se aplicarán pronto a máquinas más asequibles

Como ya hemos visto, el software integrado exige un entorno en el cual el usuario tenga acceso instantáneo a la totalidad de las diversas tareas que pueda requerir, donde los procedimientos de operatoria permanezcan constantes con independencia de la aplicación que se esté utilizando, y donde la información se pueda transferir libremente entre distintas aplicaciones. Existen muchas formas diferentes de conseguir estos objetivos.

El *Lotus 1-2-3* emplea el familiar formato de hoja electrónica, en el que las cifras y las fórmulas se entran en una cuadrícula de "celdas" y se pueden corregir libremente y volver a calcular al instante. No obstante, ofrece muchas facilidades adicionales y puede ser usado para mucho más que para la predicción y el análisis financiero. Las celdas de la hoja electrónica se pueden utilizar para almacenar información tal como nombres y números de teléfono además de datos numéricos, de modo que se podría emplear un área específica de la cuadrícula como una tabla conteniendo detalles que vengan al caso; por ejemplo, una lista de clientes y sus correspondientes números de cuenta bancaria. Dado que el *1-2-3* ofrece funciones de búsqueda y de reorganización para tal información, esta zona de la cuadrícula se puede utilizar en realidad como una pequeña base de datos. También es posible tomar un conjunto de celdas que contengan datos numéricos y emplear este paquete para visualizar esta información en forma de distintos tipos de gráficos, eliminando por consiguiente la necesidad de un programa separado de gráficos para gestión. Por último, las capacidades del *1-2-3* para tratamiento de textos permiten que se lo pueda utilizar para escribir informes, si bien las limitaciones de memoria impiden su empleo como un auténtico procesador.

Esta combinación de distintas facilidades conduce a que el *1-2-3* sea suficiente para los requerimientos de muchos usuarios. Debido a que toda la información para distintas aplicaciones está contenida en una única hoja electrónica, es fácil obtener resultados que serían imposibles con programas tradicionales. Por ejemplo, supongamos que un usuario de este paquete gestiona varios quioscos de periódicos situados en distintos puntos de una gran ciudad y necesita registrar cifras de ventas semanales, mensuales, trimestrales y anuales para cada puesto. La mejor forma de hacerlo consiste en colocar la ubicación de cada quiosco y sus cifras de ventas en una hoja electrónica. Las fórmulas se escriben de modo tal que las únicas cifras que el usuario debe modificar son los recibos semanales para cada puesto: las otras cifras se ajustan luego automáticamente.



Variaciones sinfónicas
El *Symphony*, de Lotus, consigue su integración convirtiendo toda la memoria para el usuario en una gran hoja de trabajo y permitiendo el acceso a la información almacenada a través de varias ventanas de pantalla. Éstas interpretan los datos según la función de su programa: visualización de tratamiento de textos, de base de datos, de hoja electrónica y de gráficos. Esto resuelve los problemas propios del intercambio de datos, pero exige grandes cantidades de RAM

Hasta ahora todo esto es material para una hoja electrónica estándar, pero ¿y si el usuario deseara colocar los puestos por orden de venta, de modo que el emplazamiento con mayores ventas fuera el primero en la lista? Estos puestos inicialmente se entrarían por orden alfabético, pero sería necesario reclasificarlos cada semana al recibir las nuevas cifras de ventas. Con el *Lotus 1-2-3* esto se puede hacer fácilmente y con rapidez. El propietario de los quioscos de periódicos podría desear un gráfico semanal que reflejara el rendimiento de cada uno; una secuencia de pulsaciones de teclas permitiría recuperar esta información para la hoja electrónica/base de datos, visualizarla en forma de gráfico e imprimirla.

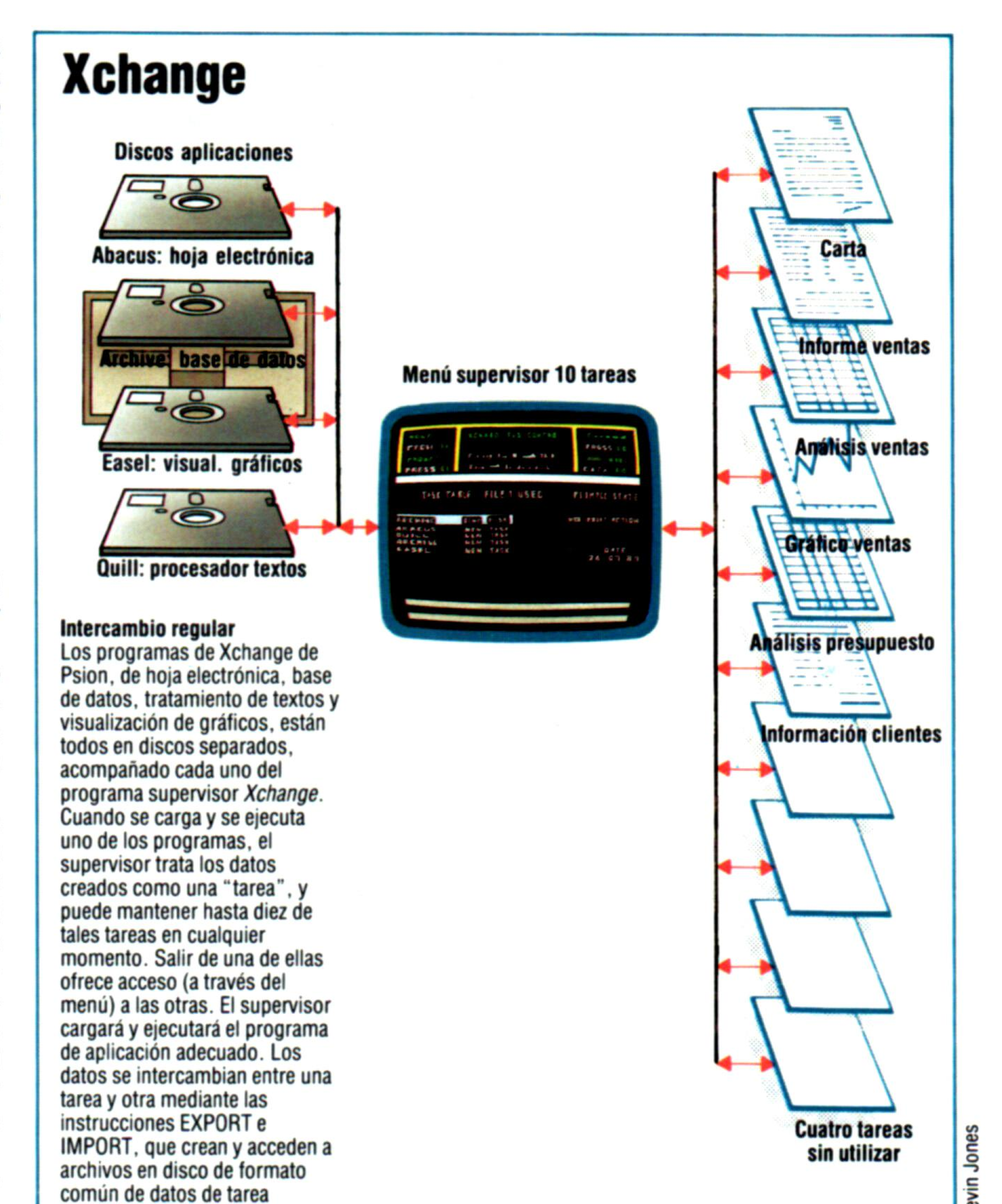
Lotus Symphony

Lotus ha continuado el concepto del *1-2-3* en el *Symphony*, que sigue el mismo principio de basar las aplicaciones en el formato de hoja electrónica. Sin embargo, el *Symphony* permite que el usuario divida la visualización en pantalla en ventanas separadas, cada una de las cuales se centra en una parte diferente de la hoja. Cada ventana está formateada de acuerdo a la información que visualiza.

Si la información a visualizar está en formato de texto, la ventana asume la forma de una pequeña pantalla para tratamiento de textos, con márgenes y posiciones de tabulación claramente marcados. Si se requiere una visualización de gráficos, la ventana muestra los ejes etiquetados y escalados. La información de base de datos se visualiza con una pantalla propia para cada entrada; esta pantalla se parece a una tarjeta de fichero. Por tanto, aunque el *Symphony* es realmente una hoja electrónica muy mejorada, da la impresión de tener cuatro grandes aplicaciones, todas en pantalla y simultáneas.

Al igual que el *1-2-3*, el *Symphony* puede "aprender" determinadas secuencias de pulsaciones de teclas, de modo que el usuario tiene la posibilidad de automatizar cualquier operación que se lleve a cabo con frecuencia. Los pequeños programas que activan la secuencia se denominan *macros de teclado*. El *Symphony* incluye asimismo su propio lenguaje de programación de alto nivel. Los programas se almacenan en la hoja de trabajo de la misma forma que todos los otros datos, y tienen acceso a todas las operaciones disponibles: por lo tanto, si el usuario debe efectuar tareas como facturación o control de stocks, puede escribir el programa en el lenguaje de programación del paquete y automáticamente pasará a formar parte de todas las aplicaciones del "entorno" *Symphony*. Una vez se haya familiarizado con el *Symphony*, le resultará más fácil escribir programas en su lenguaje propio que utilizar un lenguaje de programación independiente como el BASIC, porque el paquete ya se ocupa de ciertas tareas como dibujar gráficos o buscar y organizar datos.

El *Symphony* no es más que uno de los diversos sistemas similares que se encuentran en el mercado. El *Framework*, de Ashton Tate, es un serio competidor: proporciona una gama similar de funciones pero oculta sus estructuras de datos subyacentes en un grado aún mayor. Tanto el *Symphony* como el *Framework* son caros y exigen grandes cantidades de memoria. El primero funciona con 320 Kbytes de RAM, pero en realidad exige 512 Kbytes para aprovechar al máximo sus facilidades, mientras que el segundo necesita un mínimo de 256 Kbytes. A consecuencia de estos requerimientos, los paquetes sólo operarán en micros de 16 bits.



Intercambio regular
Los programas de Xchange de Psion, de hoja electrónica, base de datos, tratamiento de textos y visualización de gráficos, están todos en discos separados, acompañado cada uno del programa supervisor *Xchange*. Cuando se carga y se ejecuta uno de los programas, el supervisor trata los datos creados como una "tarea", y puede mantener hasta diez de tales tareas en cualquier momento. Salir de una de ellas ofrece acceso (a través del menú) a las otras. El supervisor cargará y ejecutará el programa de aplicación adecuado. Los datos se intercambian entre una tarea y otra mediante las instrucciones EXPORT e IMPORT, que crean y acceden a archivos en disco de formato común de datos de tarea

Es interesante resaltar que tanto el *Symphony* como el *Framework* funcionan con todo el paquete cargado en memoria, no requiriendo intercambio de información entre el disco y la memoria, como sucede con la mayoría de los programas de gestión. En teoría, por supuesto, la memoria de ordenador continúa volviéndose cada vez más barata, de modo que no deja de ser comprensible que quienes desarrollan software den por sentado que la mayoría de los usuarios dispondrán de ella en grandes cantidades. En la práctica, sin embargo, éste no es el caso todavía y pasará algún tiempo antes de que esta integración, que consume tanta memoria, se convierta en un lugar común. A pesar de que un programa como el *Symphony* establece nuevos estándares de rendimiento, esta clase de software aún está restringido por limitaciones de hardware; el *Symphony* consigue cumplir tan eficientemente las labores asignadas únicamente gracias a que se trata de un programa muy amplio y cuidadosamente acabado.

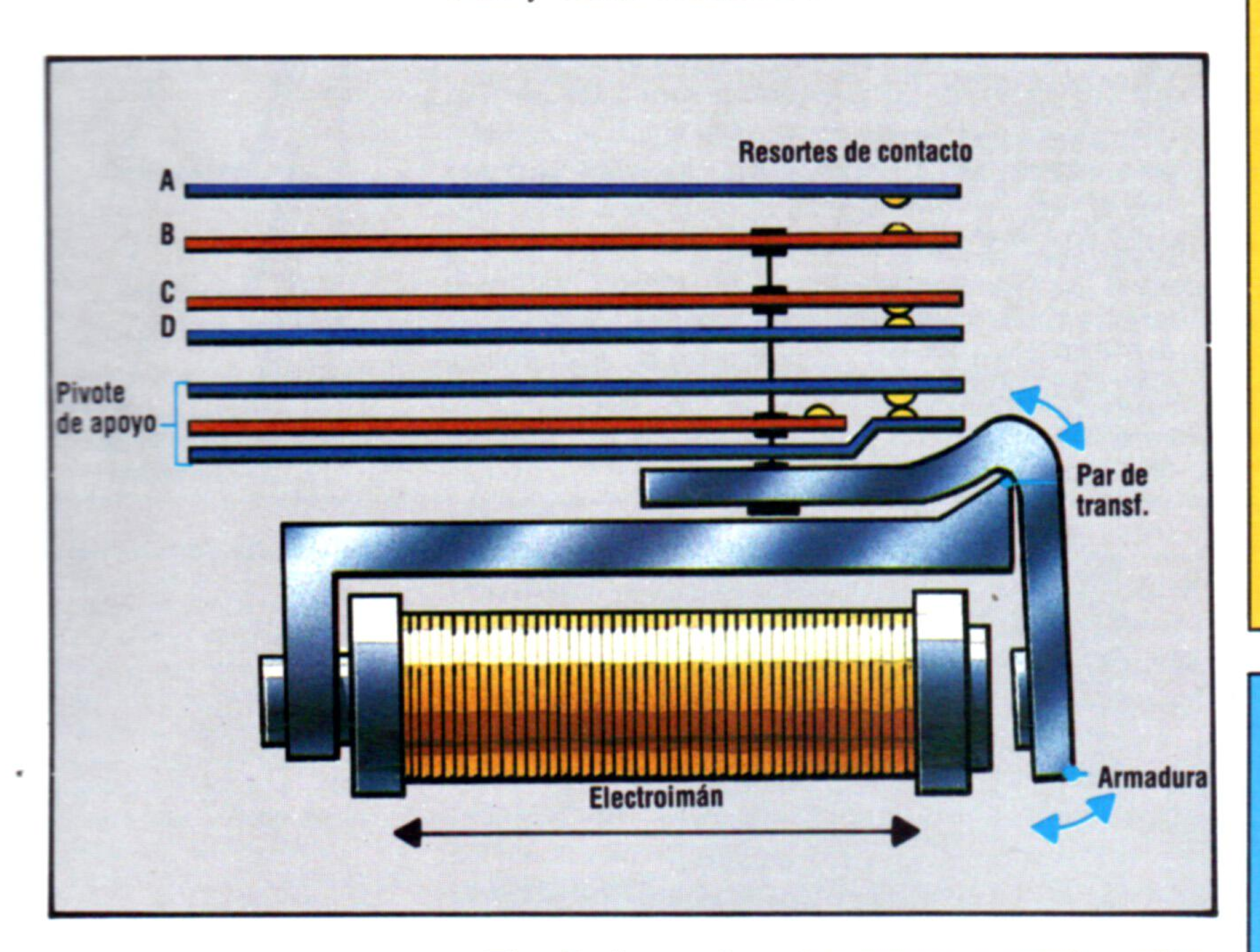
En los últimos veinte años se ha desarrollado un método alternativo para proporcionar software integrado y en la actualidad están comenzando a salir al mercado paquetes con este procedimiento.

Fuente de poder

Continuamos nuestra serie de "Bricolaje" con una explicación de cómo construir una caja de relés

Utilizando una caja de relés, un ordenador es capaz de encender y apagar las luces de una vivienda a intervalos preestablecidos. Estaría, también, en condiciones de programar una grabadora de video o de audio para que, en ausencia del usuario, le grabara una determinada emisión de televisión o radio.

Los relés eléctricos son interruptores on/off que se pueden activar mediante una señal eléctrica. En nuestra aplicación, los relés se emplean para conmutar aparatos de corriente y alto voltaje utilizando una señal de corriente y bajo voltaje. Existen muchos tipos de relés, pero el más común es el de tipo armadura, que se basa en un solenoide para abrir y cerrar conexiones.



El relé abre y cierra los contactos por la acción de pequeños movimientos de la armadura. Un voltaje adecuado aplicado a la bobina solenoide genera un campo magnético que atrae a la armadura. Cuando ésta oscila hacia la bobina, los contactos de resorte fijados en el otro extremo de la armadura se mueven verticalmente hacia arriba.

La disposición de la ilustración está en la posición "no activada", es decir, sin ningún voltaje aplicado al solenoide. En esta posición, el par de contacto AB está abierto y el par CD, cerrado. Cuando se activa el solenoide, los resortes B y C se mueven hacia arriba, haciendo que A y B se cierren y que C y D se abran. Esta disposición se podría utilizar en una de dos formas: para encender un circuito mientras se apaga el otro, o bien, más simplemente, para cerrar o abrir un circuito.

Además de esta modalidad de operación, un relé puede actuar como un mecanismo de transferencia. En el diagrama, los tres resortes inferiores están dispuestos de modo tal (en la posición no activada) que los resortes superior e inferior están en contacto. Cuando se activa el solenoide, el resorte del medio se mueve hacia arriba y hace contacto con el resorte superior, rompiendo por tanto el contacto entre los resortes superior e inferior.

Lista de componentes

Cantidad	Artículo
2	Relé 8-15 voltios de contacto de 10A 240V
2	Conector de red individual
2	Base de 29 mm
2 metros	Cable de red 3 núcleos 6 amperios
2	Enchufe de red
*	Enchufe 4 mm
*	Cable plano 2 vías 0,5 m
*	Pequeños trozos de veroboard

Nota: Los artículos señalados con un * le deben de haber sobrado de otros proyectos (véase p. 1004). Estas cantidades proporcionarán dos cajas de relés de conector único. La caja de salida puede activar cuatro de tales cajas (si así se deseara éstas podrían ser dobles) o incluso cajas de triple conector; los principios de construcción son exactamente los mismos.

Dentro de la caja

Verifique todas las conexiones para asegurar la seguridad y la continuidad, e inspeccione la placa otra vez por si hubiera puentes entre pistas. Asegúrese de que no haya comunicación entre el cable de la red y las líneas de señal.
Pegue la placa en una esquina de la base utilizando una cola de resina epoxia. Algunas colas domésticas de uso general conducen la electricidad, de modo que evítelas. Si no estuviera seguro acerca de las propiedades conductoras del adhesivo elegido, haga una prueba: coloque una delgada franja de cola sobre un trozo de cartón, déjela secar y luego conecte un tester a ambos extremos; si el medidor marca, ¡utilice otra cola! Una vez seca la cola, atornille la tapa en la caja y coloque los enchufes de 4 mm en las líneas de señal (éstas pueden ser del mismo color, porque el relé trabajará independientemente de la dirección de la corriente). Ahora conecte el enchufe de 13 A al cable de la red. El relé tiene una potencia nominal de 10 A, pero ponga en el enchufe un fusible de sólo 5 A: éste permite que se controlen los aparatos de hasta 1,2 Kw.

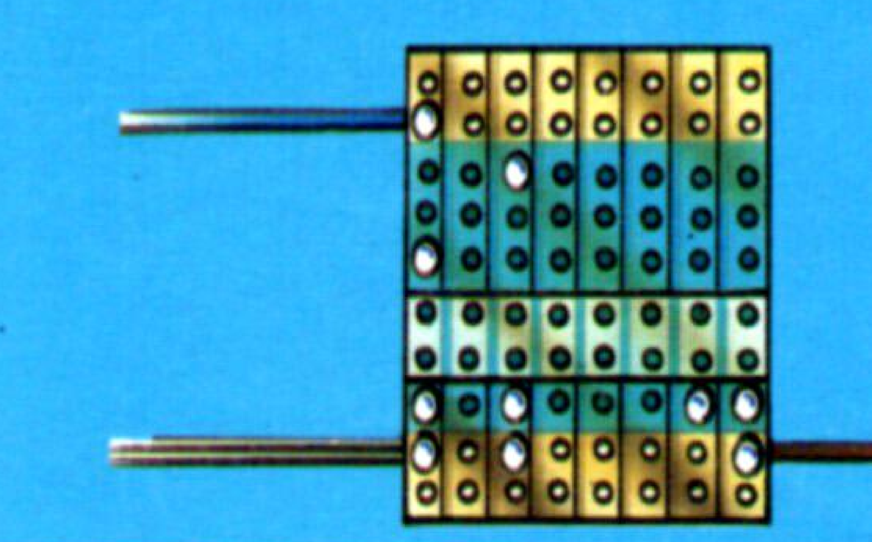


¡Atención!

Éste es un proyecto muy sencillo, pero con la electricidad debe ponerse sumo cuidado.
- **Desconecte todas las fuentes de alimentación antes de trabajar con cualquier componente.**
- **Compruebe conexiones y aislamientos con un tester antes de aplicar potencia por primera vez.**
- **Evite saltarse o abreviar etapas. Recuerde que ¡LA POTENCIA ELÉCTRICA PUEDE SER MORTAL!**

La placa de circuitos

Corte la placa según muestra la ilustración, de modo que quepa cómodamente en una esquina de la caja base. Efectúe los cortes de pistas y suéldela al relé tal como indica el diagrama.
Antes de seguir adelante verifique con suma atención la placa. Utilice el tester para comprobar posibles puentes entre pistas: ¡un error podría ser mortal!
Suelde el cable de la red marrón y el cable plano de dos vías en su lugar en el tablero. Quite una de las ranuras preformadas de la base para aceptar los cables; pero haga un nudo en ellos antes de pasarlos a través; el nudo evitará que un tirón accidental de los cables dañe la placa. Suelde a ésta un trocito de conductor de red aislado y conéctelo al terminal de tornillo "cargado" del conector. Conecte los cables azul y amarillo-verde de la red a los terminales neutral y a tierra, respectivamente



Programa de prueba

Después de construida la caja de relés y comprobadas todas las conexiones, podemos probarla escribiendo un programa para encender y apagar un dispositivo que funcione con la corriente de la red. Éste podría ser una lámpara de mesa. Ésta se debe enchufar al conector de fuente de alimentación de la caja de relés, conectando los cables de señal a los terminales positivo y negativo de la línea 0 de la caja de salida de bajo voltaje. Los cables de señal se pueden conectar en cualquiera de los terminales sin afectar el funcionamiento del relé. El cable de la red de la caja del relé se conecta, entonces, en un enchufe de la pared. Después de hechas las conexiones, entre este programa: encienda la lámpara durante cinco segundos y después la apaga.

```
10 REM PRUEBA RELE DE LA RED
20 RDD=&FE62:REGDAT=&FE60
30 ?RDD=255:REM TODAS SALIDA
40 ?REGDAT=1:REM ENCENDER LA LUZ
50 TIEMPO=0: REM ESTABLECER TEMPORIZADOR
60 REPEAT
70 UNTIL TIEMPO>500
80 ?REGDAT=0:REM APAGAR LA LUZ
```

```
10 REM PRUEBA RELE DE LA RED CBM 64
20 RDD=56579:REGDAT=56577
30 POKE RDD,255:REM TODAS SALIDA
40 POKE REGDAT,1:REM ENCENDER LA LUZ
50 T=TI:REM ESTABLECER TEMPORIZADOR
60 IF (TI-T)<300 THEN 60
70 POKE REGDAT,0:REM APAGAR LA LUZ
```

Si después de ejecutar el programa la lámpara no se encendiera, desenchufe la caja de relés de la red antes de comprobar las conexiones.

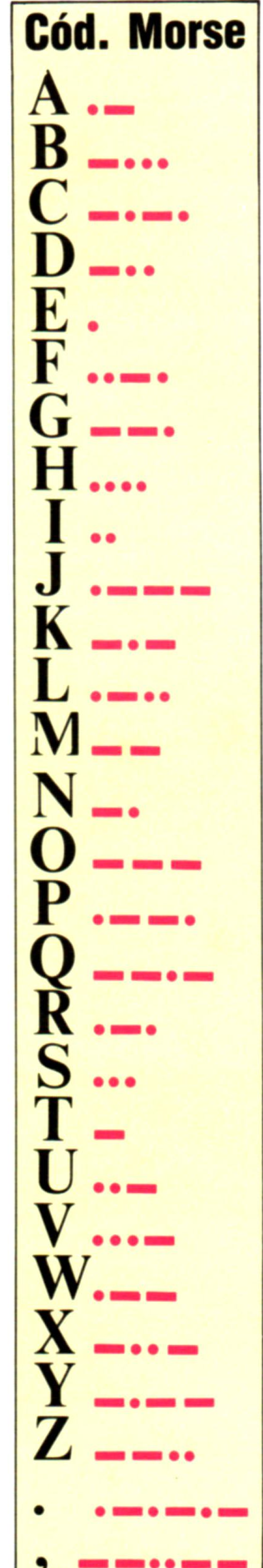
Diagrama de bloque

Así como la caja buffer aísla al ordenador de las corrientes de bajo voltaje que conmuta la caja de salida, esta caja de relés aísla al ordenador de la corriente de la red. Éste envía corriente a través de un relé de red, que entonces enciende y apaga la fuente de alimentación eléctrica. La única conexión entre la fuente de alimentación y el ordenador es el campo magnético del relé.

Proyecto de código Morse

```
10 REM ****************************************
11 REM*          PRACTICA MORSE CBM 64         *
12 REM*         ENCHUFE UNA LAMPARA EN         *
13 REM*                EL RELE                 *
14 REM*         ENTRE CUALQUIER SERIE          *
15 REM*        ALFA, Y SE REPRODUCIRA          *
16 REM*         EN MORSE CON BEEP Y            *
17 REM*           LUZ INTERMITENTE             *
20 REM ****************************************
100 GOSUB 2000:                     REM INICIO
150 FOR L=0 TO 1 STEP 0
200 PRINT“ENTRE SU MENSAJE”
220 PRINT“DIGITE ‘ADIOS’ PARA SALIR”
240 INPUT“MENSAJE”;MS$
300 ML=LEN(MS$):M$=“”
320 FOR K=1 TO ML
330 C$=MID$(MS$,K,1)
340 IF C$=>“A”ANDC$<=“Z” THEN M$=M$+C$
350 IF C$=“ ” THEN M$=M$+C$
360 NEXT K: IF M$=“” THEN NEXT L
400 ML=LEN(M$)
420 FOR K=1 TO ML
440 CH$=MID$(M$,K,1):CH=ASC(CH$)-64
450 IF CH=-32 THEN FORPP=1TO6*DE:NEXT PP
460 IF CH<>-32 THEN GOSUB 3000
480 FOR PP=1 TO 3*DE:NEXT PP
500 NEXT K
550 IF M$=“ADIOS” THEN L=1
600 NEXT L
900 END
2000 REM *************** INICIAL ***************
2100 DIM M$(26)
2110 RDD=56579:REGDAT=56577:POKE RDD,255
2120 DE=25:RX=2*DE
2130 V=54296:LF=54272:HF=54273:W=54276
2140 A=54277:S=54278
2150 FOR K=LF TO LF+24:POKEK,0:NEXT K
2160 POKEA,24:POKES,129:POKEV,15
2200 DATA “.-”,“-...”,“-.-.”,“-..”,“.”
2220 DATA “..-.”,“--.”,“....”,“..”
2240 DATA “.---”,“-.-”,“.-..”,“--”
2260 DATA “-.”,“---”,“.--.”,“--.-”
2280 DATA “.-.”,“...”,“-”,“..-”,“...-”
2300 DATA “.--”,“-..-”,“-.--”,“--..”
2400 FOR K=1 TO 26: READ M$(K):NEXT K
2990 RETURN
3000 REM ************* FLASH & BEEP *************
3050 PRINT CH$,M$(CH)
3100 N=LEN(M$(CH))
3200 FOR C=1 TO N
3220 D=DE-(ASC(MID$(M$(CH),C,1))-46)*RX
3240 POKE REGDAT,1            :REM FLASH
3250 POKE LF,172:POKE HF,57:REM BEEP
3260 POKE W,33:FOR PP=1 TO D:NEXT PP
3270 POKE W,32
3280 POKE LF,0:POKE HF,0      :REM QUITAR BEEP
3290 POKE REGDAT,0            :REM QUITAR FLASH
3300 FOR PP=1 TO DE:NEXT PP
3320 NEXT C
3490 RETURN
```

```
10 REM ****************************************
11 REM*           PRACTICA MORSE BBC           *
12 REM*         ENCHUFE UNA LAMPARA EN         *
13 REM*            EL RELE DE RED:             *
14 REM*         ENTRE CUALQUIER SERIE          *
15 REM*        ALFA, Y SE REPRODUCIRA          *
16 REM*         EN MORSE CON BEEP Y            *
17 REM*           LUZ INTERMITENTE             *
20 REM ****************************************
100 PROCinicializar
120 FIN=FALSE
150 REPEAT
200 PRINT“ENTRE SU MENSAJE”
220 PRINT“TECLEE ‘ADIOS’ PARA SALIR”
240 INPUT“MENSAJE”,MS$
300 ML=LEN(MS$):M$=“”
320 FOR K=1 TO ML
330 C$=MID$(MS$,K,1)
340 IF C$>=“A” ANDC$<=“Z” THEN M$=M$+C$
350 IF C$=“ ” THEN M$=M$+C$
360 NEXT K:IF M$=“” THEN UNTIL FALSE
400 ML=LEN(M$)
420 FOR K=1 TO ML
440 CH$=MID$(M$,K,1):CH=ASC(CH$)-64
450 IF CH=-32 THEN PROCdemora(6*DE*IX)
460 IF CH<>-32 THEN PROCbeepflash
480 PROCdemora(3*DE)
500 NEXT K
550 IF M$=‘ADIOS’ THEN FIN=TRUE
600 UNTIL FIN
900 END
2000 REM *************** INICIO ***************
2050 DEF PROCinicializar
2100 DIM M$(26)
2110 RDD=&FE62:REGDAT=&FE60:?RDD=255
2120 DE=3:RX=2*DE:IX=30
2200 DATA “.-”,“-...”,“-.-.”,“-..”,“.”
2220 DATA “..-.”,“--.”,“....”,“..”
2240 DATA “.---”,“-.-”,“.-..”,“--”
2260 DATA “-.”,“---”,“.--.”,“--.-”
2280 DATA “.-.”,“...”,“-”,“..-”,“...-”
2300 DATA “.--”,“-..-”,“-.--”,“--..”
2400 FOR K=1 TO 26:READ M$(K):NEXT K
2990 ENDPROC
3000 REM ************* FLASH & BEEP *************
3020 DEFPROCbeepflash
3050 PRINT CH$,M$(CH)
3100 N=LEN(M$(CH))
3200 FOR C=1 TO N
3220 D=DE-(ASC(MID$(M$(CH),C,1))-46)*RX
3240 ?REGDAT=1                     :REM FLASH
3260 SOUND 1,-15,200,D             :REM BEEP
3270 PROCdemora(IX*D)
3290 ?REGDAT=0                     :REM QUITAR FLASH
3300 PROCdemora(DE*IX)
3320 NEXT C
3490 ENDPROC
4000 REM *************** DEMORA ***************
4100 DEFPROCdemora(tiempo)
4200 FOR DD=1 TO tiempo:NEXT DD
4490 ENDPROC
```



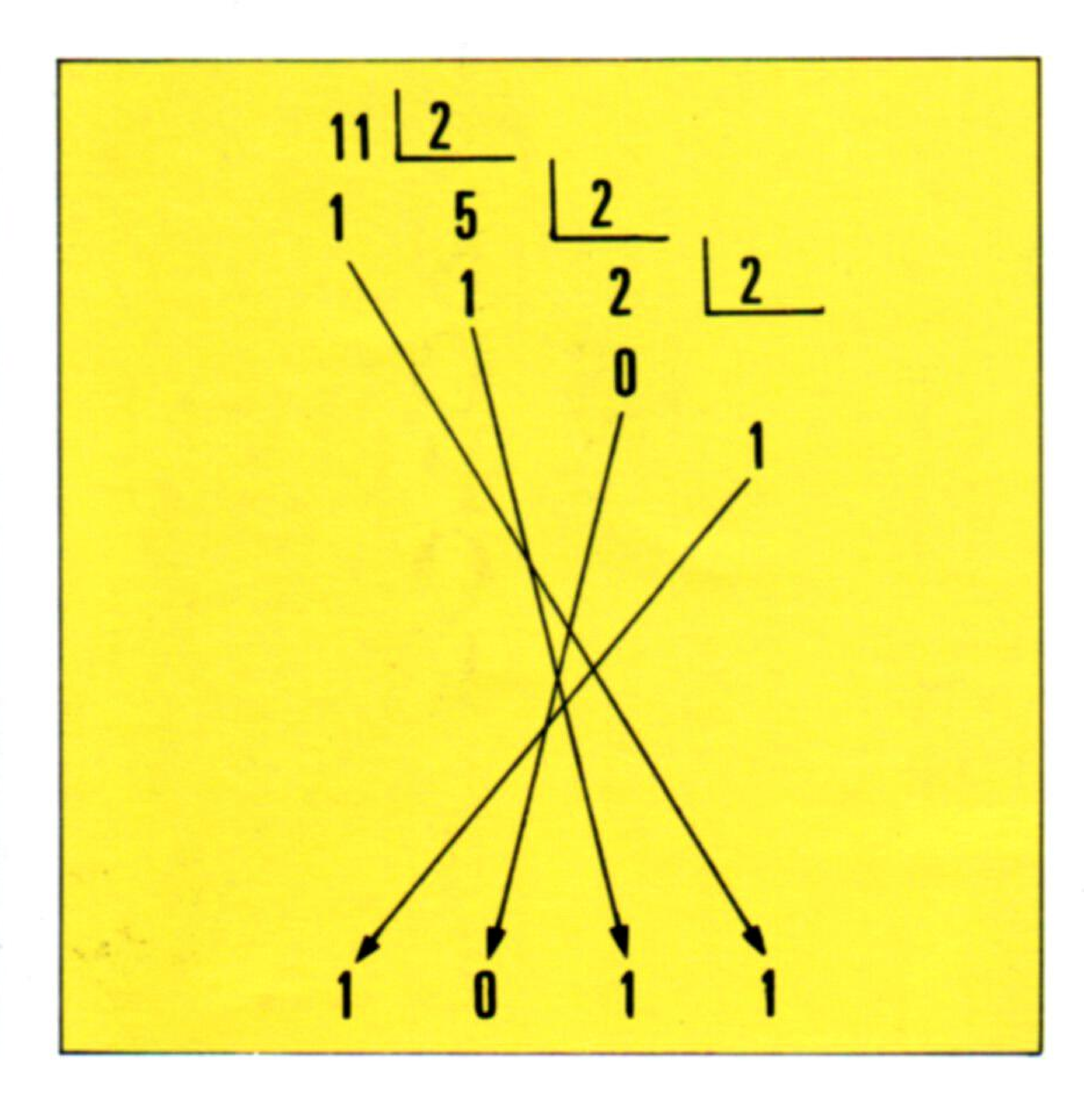




De decimal a binario

En este último capítulo de este apartado desarrollaremos un diseño que permite convertir una cantidad decimal en su correspondiente número binario

Para realizar la operación de convertir un número decimal en binario se va dividiendo la cantidad decimal entre la base del sistema a que se quiere convertir (en este caso, binario=base 2), hasta que el dividendo ya no contenga al divisor, es decir, hasta que no se pueda dividir más por la base. Entonces se toma el último cociente y todos los restos, en orden inverso a como han aparecido, o sea de derecha a izquierda en orden ascendente. Veamos un ejemplo: $11_{(10}=1011_{(2}$



De este modo, se ve la necesidad de guardar cada uno de los restos que surjan. Para ello se dejarán en una tabla que hemos creado con capacidad para 16 elementos; así podrán representarse números con un valor máximo de 65 535.

```
  5 REM CONVERSION BINARIA
 10 DIM T(20)
 20 INPUT "ENTRAR VALOR DECIMAL";N
 30 E=1
 40 IF N=1 THEN GOTO 100
 50 IF N=INT(N/2)*2 THEN T(E)=0 : GOTO 70
 60 T(E)=1
 70 E=E+1
 80 N=INT(N/2)
 90 GOTO 40
100 T(E)=1
110 FOR X=E TO 1 STEP-1
120 PRINT T(E);
130 NEXT X
140 END
```

INICIO

DIMENSIONAR TABLA T DE 16 ELEMENTOS

NÚMERO N

ÍNDICE =1

NÚMERO =1 ?

SÍ

TABLA SEGÚN ÍNDICE=1

DESDE X ÍNDICE HASTA 1

STEP–1

TABLA SEGÚN ÍNDICE X

SIGUIENTE

FIN

NO

N DIVISIBLE 2 ?

NO

TABLA SEGÚN ÍNDICE=1

SÍ

TABLA SEGÚN ÍNDICE=0

SUMAR 1 A ÍNDICE

DIVIDIR EL NÚMERO POR 2

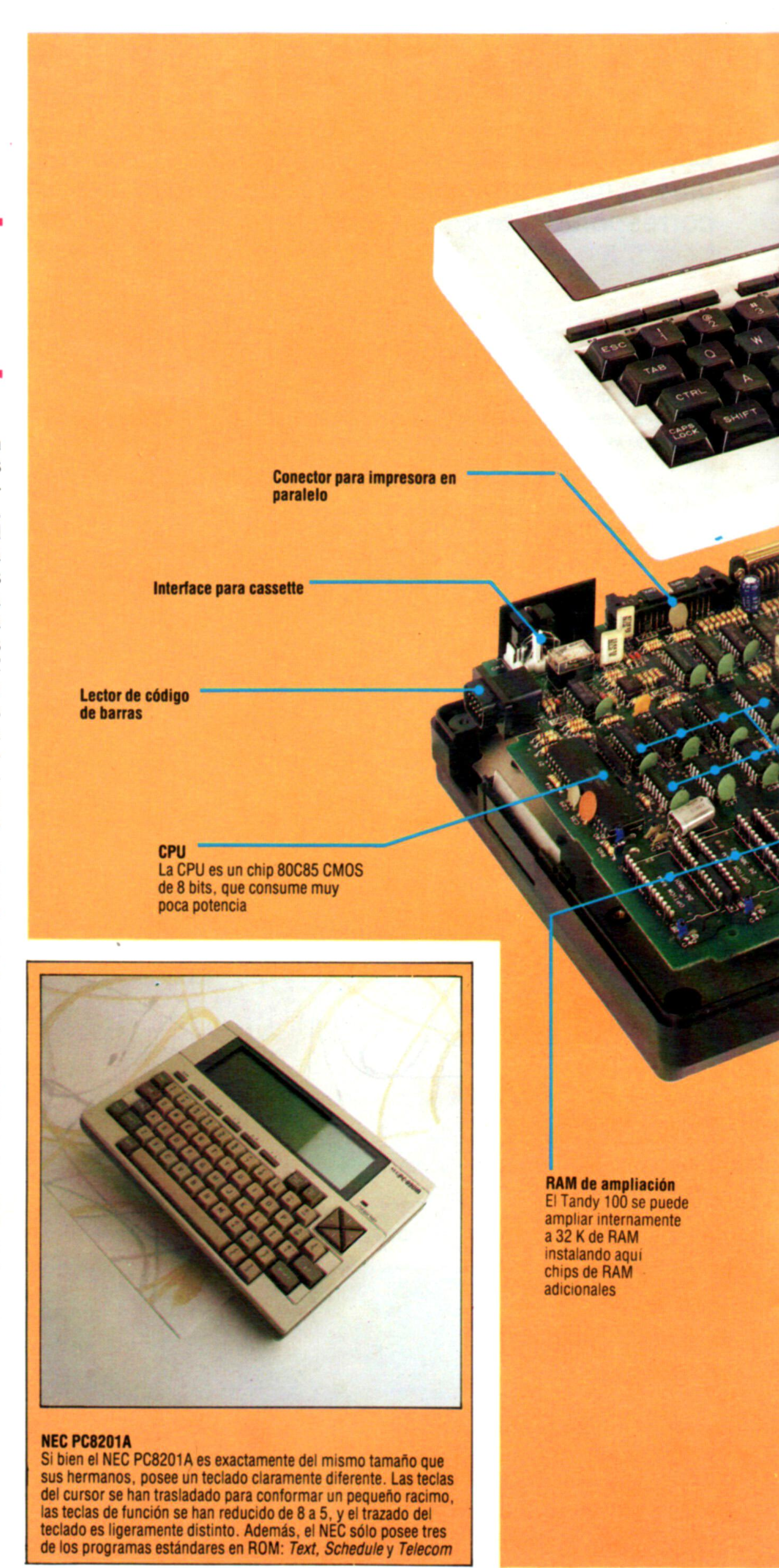
Un trío de calidad

Demos una mirada a una máquina "de regazo", el Tandy Modelo 100, y comparémosla con dos dignos competidores

El proceso por el cual un fabricante adquiere un producto acabado, modifica algunos elementos para darle un aspecto exclusivo y lo empaqueta después como un artículo "fabricado para el cliente", se conoce como *ingeniería del distintivo*. En el campo de la electrónica de consumo, esta técnica ya existe desde hace tiempo, con productos tales como televisores y equipos de alta fidelidad. La misma técnica se está comenzando a aplicar ahora en el mercado de ordenadores, y tres populares máquinas portátiles (el Tandy Modelo 100, el NEC PC8201A y el Olivetti M10) son producto de un acuerdo de este tipo. La firma japonesa Kyocera es la que fabrica los tres ordenadores y se los vende a Tandy, NEC y Olivetti, que ponen su propia carcasa a las máquinas y las comercializan bajo su respectivo nombre. En este capítulo analizaremos el Tandy 100 y pondremos de relieve las diferencias entre esta máquina y sus hermanas.

Pesando poco menos de 1,8 kg, los modelos Tandy, NEC y Olivetti se inscriben claramente en la categoría de máquinas "de regazo". El Modelo 100 tiene un teclado estilo QWERTY completo, software incorporado basado en ROM y una pantalla en cristal líquido (LCD) a pilas. Puede funcionar completamente con la energía de éstas, y el contenido de la RAM no se pierde cuando se apaga la máquina. Los archivos se pueden almacenar en RAM y acceder directamente como si la memoria estuviera en cassette o en disco. El Modelo 100 también se puede conectar a una cassette o unidad de disco para almacenamiento externo, pero la memoria permanente hace que almacenar datos importantes "sobre la marcha" resulte sencillo.

La pantalla LCD proporciona ocho líneas de 40 caracteres y ofrece la posibilidad de mezclar texto y gráficos. La visualización se compone de 15 360 puntos, cada uno de los cuales se puede referenciar individualmente. Los caracteres se forman en una matriz de 6 por 8 y se pueden visualizar caracteres en mayúscula y en minúscula. El Modelo 100 incorpora un juego completo de caracteres internacionales, así como otro especial de caracteres para gráficos, a diferencia de la máquina NEC, que sólo posee tres caracteres para gráficos. Tanto el NEC como el Tandy tienen pantallas LCD que yacen extendidas sobre sus carcasas, pero el Olivetti M10 configura una pantalla móvil que se puede inclinar para obtener un ángulo de trabajo cómodo, proporcionando, por consiguiente, mayor flexibilidad. El NEC y el Tandy tienen mandos de contraste



Conector para impresora en paralelo

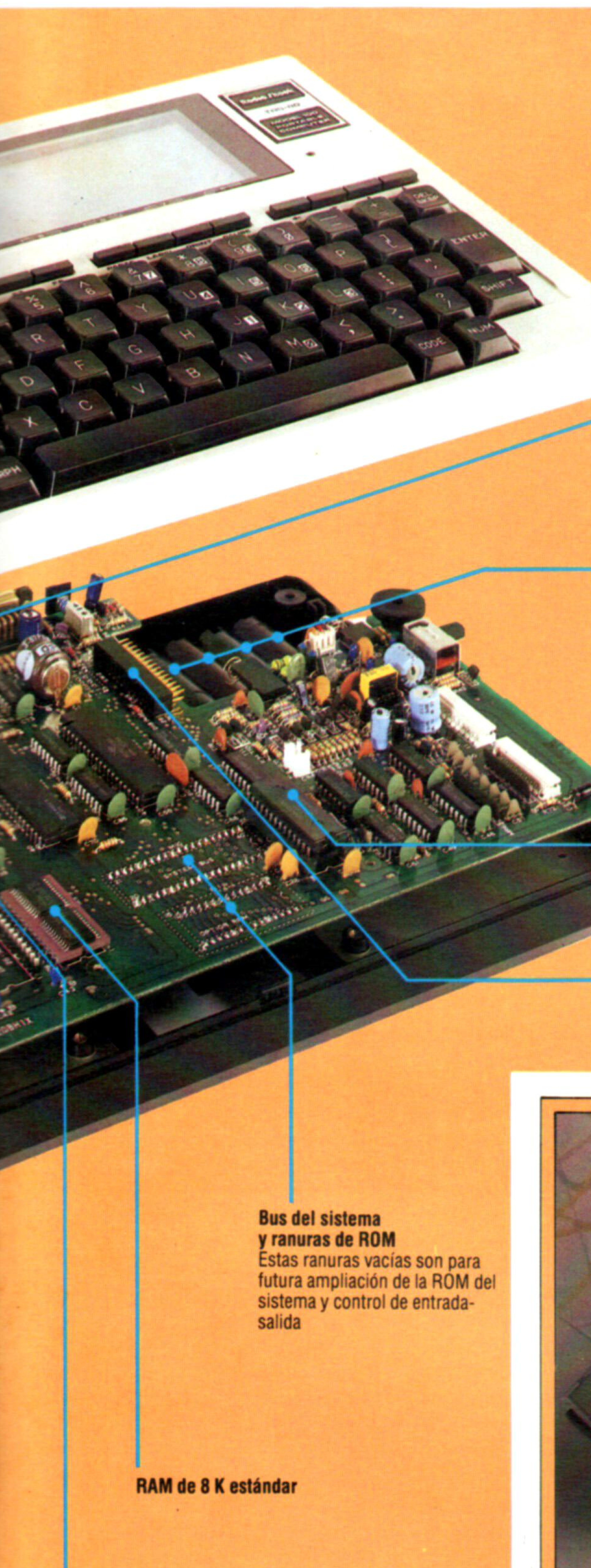
Interface para cassette

Lector de código de barras

CPU
La CPU es un chip 80C85 CMOS de 8 bits, que consume muy poca potencia

RAM de ampliación
El Tandy 100 se puede ampliar internamente a 32 K de RAM instalando aquí chips de RAM adicionales

NEC PC8201A
Si bien el NEC PC8201A es exactamente del mismo tamaño que sus hermanos, posee un teclado claramente diferente. Las teclas del cursor se han trasladado para conformar un pequeño racimo, las teclas de función se han reducido de 8 a 5, y el trazado del teclado es ligeramente distinto. Además, el NEC sólo posee tres de los programas estándares en ROM: *Text*, *Schedule* y *Telecom*



Conector para modem
Ésta es una puerta RS232 estándar para comunicaciones en serie. El software para telecomunicaciones está incorporado en la máquina

Fuente de alimentación
El Modelo 100 funciona hasta 20 horas con 4 pilas alcalinas "AA". La memoria interna se conserva hasta 30 días en virtud de pilas recargables de níquel-cadmio, que se recargan automáticamente cuando está conectada a la red

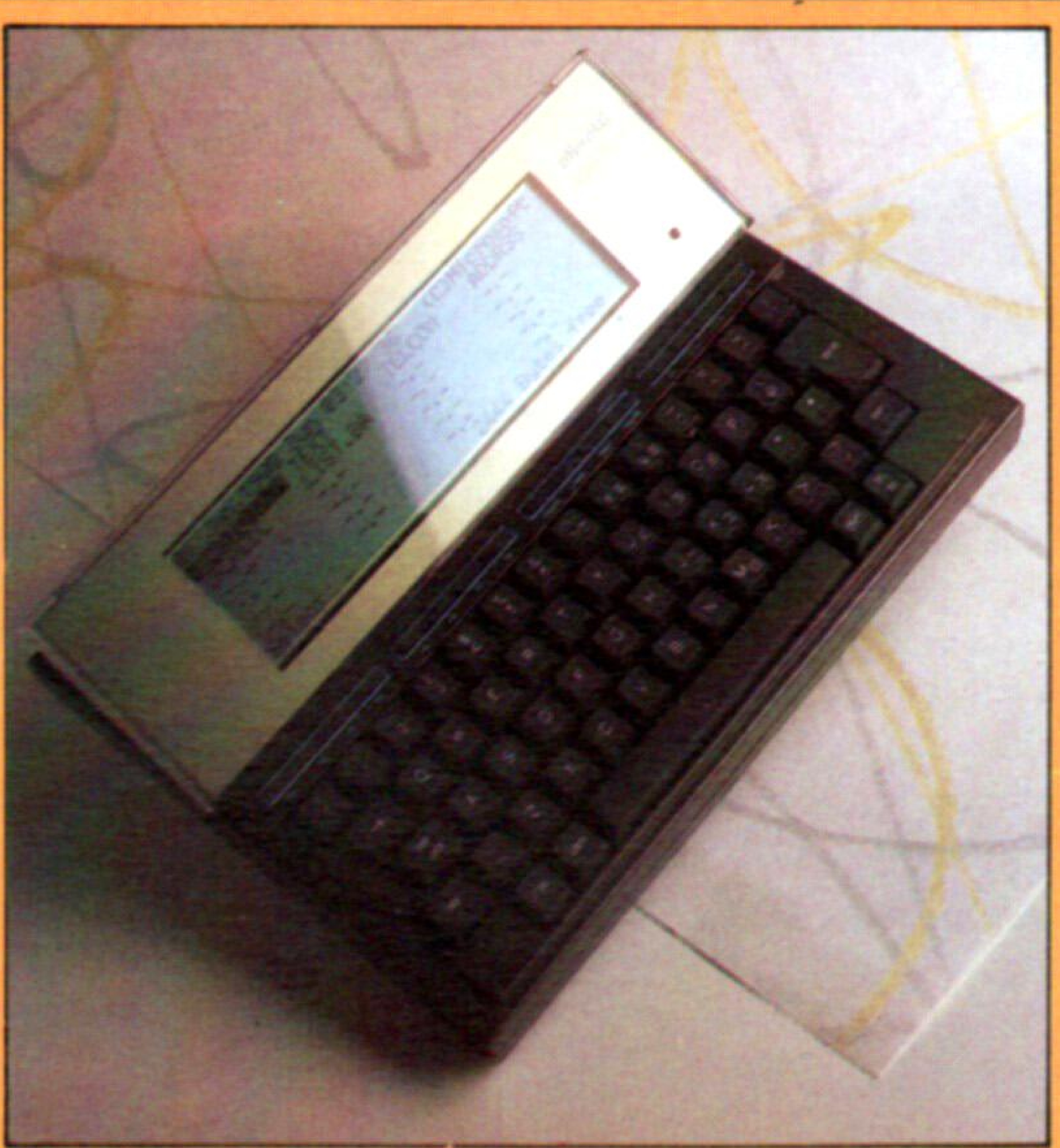
ROM estándar
Este chip contiene el BASIC Microsoft y el software incorporados

Conector LCD
Un cable plano conecta la pantalla LCD con la placa del sistema

Bus del sistema y ranuras de ROM
Estas ranuras vacías son para futura ampliación de la ROM del sistema y control de entrada-salida

RAM de 8 K estándar

Unidad de teclado
Estos chips controlan la entrada y la salida del teclado y contienen los juegos de caracteres



Olivetti M10
La versión Olivetti de esta máquina posee un interesante detalle exclusivo de ella: la pantalla LCD se puede inclinar hasta un ángulo de unos 40°, lo que hace que la visualización resulte más fácil de leer. Posee el mismo teclado que el Tandy 100 y los cinco paquetes de software estándares basados en ROM

TANDY MODELO 100

DIMENSIONES
300 × 215 × 50 mm

CPU
CMOS 80C85 de ocho bits, 2,4 MHz

MEMORIA
RAM de 8 K o 24 K, ampliable en bloques de 8 K hasta un total de 32 K; 32 K de ROM, incluyendo software y BASIC Microsoft

PANTALLA
LCD de 40 columnas × 8 líneas; gráficos de 240 × 64 puntos direccionables; caracteres ASCII e internacionales, 39 caracteres para gráficos

INTERFACES
Impresora en paralelo, cassette, puerta en serie RS232, lector de código de barras, bus del sistema

LENGUAJES DISPONIBLES
BASIC Microsoft

TECLADO
Teclado estándar tipo máquina de escribir, de 56 teclas; teclado numérico incorporado; 8 teclas de función programable; 4 teclas de instrucciones y 4 teclas de control de cursor

DOCUMENTACION
Guía de referencia rápida al BASIC de 48 páginas; detallado manual de operatoria de 200 páginas

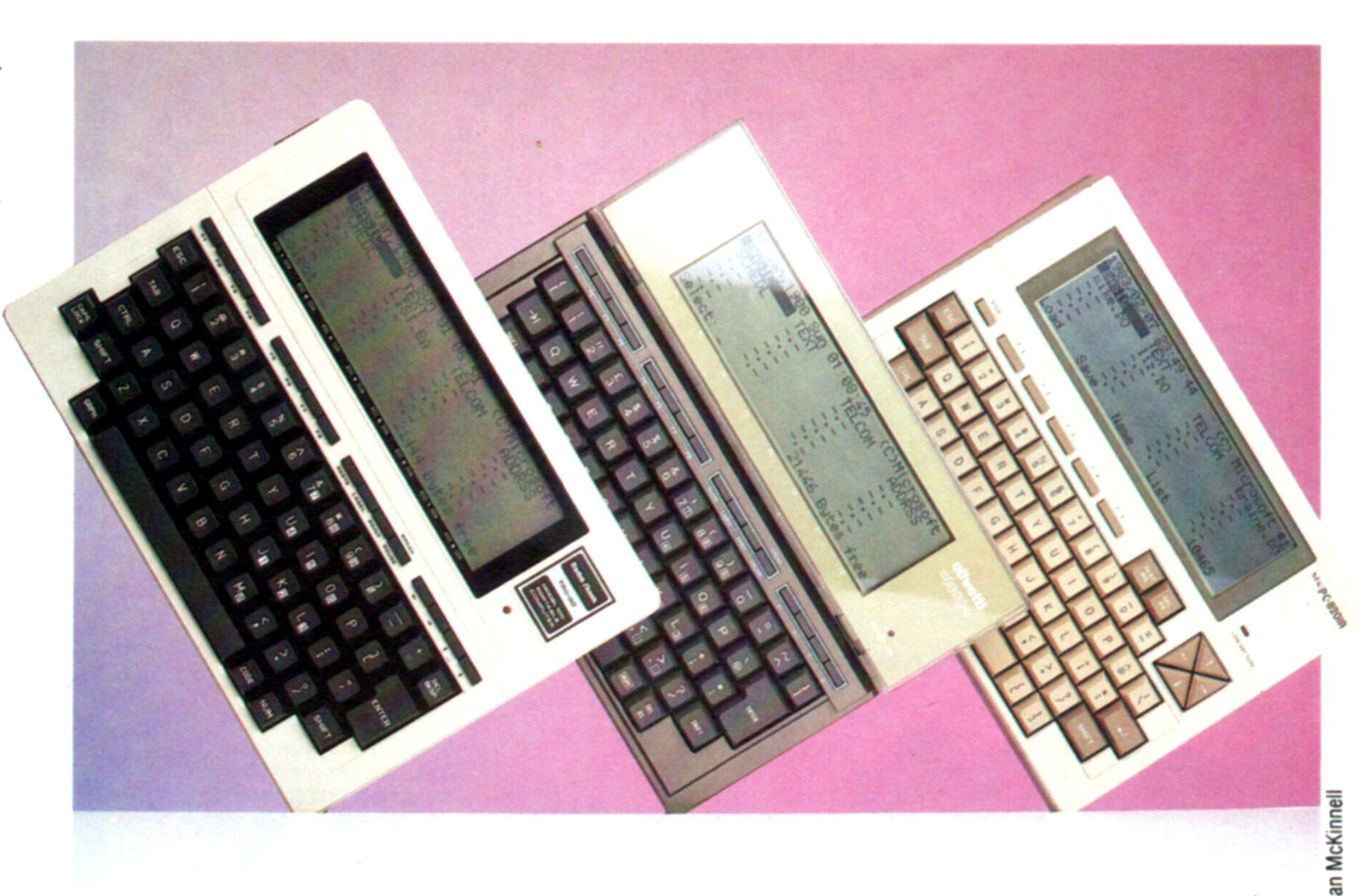
VENTAJAS
El Tandy 100 es pequeño y, sin embargo, ofrece la mayoría de las características necesarias para la informática 'seria'. La memoria permanente y el funcionamiento a pilas lo hacen muy portátil

DESVENTAJAS
Las limitaciones de la RAM reducen el uso práctico de la máquina a aplicaciones portátiles. No tendría capacidad para 'superar' su condición de portátil y funcionar como un genuino ordenador de sobremesa

DIFERENCIAS
Olivetti M10: Teclado de 57 teclas; 47 caracteres para gráficos; unidad de pantalla inclinable; un manual para el usuario
NEC PC8201A: 57 teclas; rosa de cursor; 5 teclas de función; 16 K de RAM ampliables a 96 K; sólo tres caracteres para gráficos

Ian McKinnell

Cortesía de CL Projects



De tal palo tal astilla
A pesar de ser distribuidos por tres empresas diferentes, el Tandy Modelo 100, el Olivetti M10 y el NECP PC8201A son todos fabricados por la misma empresa, la Kyocera japonesa. Existen algunas pequeñas diferencias entre las tres máquinas, pero hasta un ojo inexperto puede ver que los tres ordenadores portátiles comparten una herencia común

Ian McKinnell

ajustables para conferirle más claridad a la pantalla.

El excelente teclado del Tandy posee teclas especiales para acceder a los gráficos incorporados o para cambiar varias de las teclas alfabéticas convirtiéndolas en un teclado numérico. Utilizando esta facilidad, la tecla M se convierte en 0; J, K y L se transforman en 1, 2 y 3; por su parte, U, I y O se convierten en 4, 5 y 6; finalmente, 7, 8 y 9 conservan su función normal. Las tres máquinas poseen cuatro teclas para el cursor, pero la posición de las mismas varía. Los modelos Tandy y Olivetti poseen cuatro pequeñas teclas, una junto a la otra, situadas arriba y a la derecha del teclado; el NEC PC8201A posee un teclado de cursor, con las cuatro teclas para el cursor formando un cuadrado.

Las máquinas incorporan, asimismo, teclas de función programable, que se utilizan con el software incorporado para dirigir las funciones de tratamiento de archivos y el movimiento en y entre los programas retenidos en ROM. También existen algunas diferencias en este aspecto. El Modelo 100 de Tandy tiene ocho teclas de función, más cuatro teclas adicionales que se utilizan para llevar a cabo tareas internas. PASTE se emplea para trasladar datos de un programa a otro; LABEL les asigna nombres a las teclas de función de modo que el usuario siempre sabe qué función realiza cada tecla; PRINT envía archivos directamente a la impresora, y la tecla BREAK interrumpe la ejecución del programa. Este trazado se repite en el Olivetti, pero el NEC posee cinco teclas de función, programables para un total de 10 funciones, y una tecla Pause.

El Modelo 100 y el M10 se suministran con 8 o 24 K de RAM, que se pueden ampliar a 32 con la adición de un paquete de RAM interna. El NEC es ligeramente diferente: se suministra con 16 K, pero se puede ampliar internamente a 64 o a 96 K si se utiliza la puerta para ampliación incorporada.

El Modelo 100 viene con BASIC Microsoft y un pequeño sistema de "administración doméstica" que dirige el software interno. Al conectarse a la red, se visualizan los archivos almacenados en memoria, junto con los títulos de los programas internos suministrados.

Los programas proporcionados incluyen *Text*, un pequeño procesador de textos apto para apuntar memorándums o escribir cartas o informes cortos; éste es especialmente adecuado para tomar notas y muy útil para periodistas, estudiantes o usuarios de gestión. *Schedule* es un pequeño programa de base de datos, diseñado específicamente como ayuda en el registro de citas, gastos, "cosas que hacer" y otros recordatorios. Una función de búsqueda incorporada facilita la localización rápida de información. Un tercer programa, llamado *Address*, constituye una pequeña base de datos similar que puede parecer innecesaria, dada la existencia de *Schedule*. Por último, hay un programa de comunicaciones basado en RS232 denominado *Telecom*, que permite conectar el Modelo 100 con un modem para comunicaciones telefónicas: con unas pocas pulsaciones de teclas se pueden enviar o recibir datos desde remotos ordenadores. El NEC PC8201A sólo viene con BASIC, *Text* y *Telecom*.

Las tres máquinas están bien provistas de interfaces, disponiendo cada una de una puerta para comunicaciones RS232, una puerta para impresora en paralelo, interface para cassette y un conector para lector de código de barras. Los modelos Tandy y Olivetti incluyen un bus del sistema, mientras que el NEC agrega a su relación de interfaces dos puertas en serie adicionales.

La utilización de una máquina básica, con ligeras diferencias entre los tres modelos distintos, ha significado que los fabricantes puedan proporcionar productos de alta calidad sin que ninguna de las empresas haya tenido que hacer frente sola a todos los costos de desarrollo. Sólo nos resta resaltar que cualquiera de estos tres ordenadores portátiles ofrece una buena relación calidad-precio.



¿Qué motocicleta?

He aquí las versiones del juego del capítulo anterior para otros dos ordenadores: Commodore 64 y BBC Micro

A diferencia de las versiones de BASIC que utilizan el Spectrum y el BBC, el BASIC del Commodore 64 no posee ninguna instrucción que nos permita activar pixels individuales. En la versión del juego que ofrecemos aquí, utilizamos caracteres en baja resolución para dibujar el sendero de las "motos luminosas". Se emplea un carácter espacio invertido, con POKE código 160: para trazar este carácter en la pantalla tenemos que colocar (POKE) este valor en el mapa de pantalla de la memoria y especificar el color en la posición correspondiente.

Al igual que la versión para el Spectrum, el juego para el Commodore no está estructurado para obtener la máxima velocidad de ejecución. En aquellos puntos en los que la velocidad carezca de importancia se introduce algo de estructuración en forma de llamadas a subrutinas para incrementar el marcador y dotar de intermitencia a la pantalla.

Debido a que el BASIC BBC opera considerablemente más rápido que el BASIC Spectrum o Commodore y permite llamar a módulos estructurados a modo de procedimientos, la versión del juego para el BBC es muy estructurada. La mayoría de las versiones de BASIC permiten la estructuración mediante el empleo de subrutinas, pero ello reduce la velocidad de ejecución, puesto que se debe efectuar una búsqueda cada vez que se las llama. El BASIC BBC toma nota de la posición de un procedimiento cuando el mismo es llamado por primera vez, y la almacena en una tabla de referencia.

Commodore 64

```
10 REM CBM 64
15 POKE53281,0:POKE53280,4:REM COLOR PANT./TABL.
20 SC=1024:C0=55296
25 PRINTCHR$(147):REM LIMPIAR PANTALLA
30 X1=2:Y1=12:X2=37:Y2=12
35 DX=1:DM=-1:DN=0:DY=0
40 PRINTCHR$(19)CHR$(158)"JUGADOR 1:"S1
50 PRINTCHR$(19)TAB(27)"JUGADOR 2:"S2
52 FORI=1TO8:PRINTCHR$(17);:NEXT
54 PRINTTAB(14)"PULSE UNA TECLA"
55 GETJ$:IFJ$<>""THEN55
56 GETA$:IFA$=""THEN56
57 PRINTTAB(14)CHR$(145)"
60 REM BUCLE PRINCIPAL
70 GETA$
80 IFA$="W"THENDY=-1:DX=0
90 IFA$="X"THENDY=1:DX=0
100 IFA$="A"THENDX=-1:DY=0
110 IFA$="D"THENDX=1:DY=0
120 IFA$="0"THENDN=-1:DM=0
130 IFA$="M"THENDN=1:DM=0
140 IFA$="J"THENDM=-1:DN=0
150 IFA$="L"THENDM=1:DN=0
155 Y1=Y1+DY
156 IFY1<1ORY1>24THENF=0:GOSUB1000:GOTO25
157 X1=X1+DX
158 IFX1<0ORX1>39THENF=0:GOSUB1000:GOTO25
160 Y2=Y2+DN
162 IFY2<1ORY2>24THENF=1:GOSUB1000:GOTO25
164 X2=X2+DM
166 IFX2<0ORX2>39THENF=1:GOSUB1000:GOTO25
167 P1=X1+40*Y1
168 P2=X2+40*Y2
170 IFPEEK(SC+P1)=160THENF=0:GOSUB1000:GOTO25
180 IFPEEK(SC+P2)=160THENF=1:GOSUB1000:GOTO25
190 POKE SC+P1,160
200 POKE CO+P1,14
210 POKE SC+P2,160
220 POKE CO+P2,5
230 GOTO70:REM RECOMENZAR BUCLE
240 :
1000 REM S/R MARCADOR
1020 IF F=1THENS1=S1+1:GOSUB2000:RETURN
1030 S2=S2+1:GOSUB2000:RETURN
1999 :
2000 REM S/R FLASH PANTALLA
2010 FORJ=1TO10
2020 FORI=0TO15
2030 POKE53281,I
2040 NEXTI,J
2045 POKE53281,0
2050 RETURN
```

BBC Micro

```
10 REM BBC
20 MODE1:moto1=0:moto2=0
30 PROCinicializar
40 PROCtecla
50 PROCborde
60 PROCmarcador
70 PROCtrazar
80 PROCteclado
90 PROCleer—punto
100 GOTO70
110 END
120 DEF PROCborde
130 GCOL0,borde
140 MOVE0,0
150 DRAW1279,0
160 DRAW1279,980
170 DRAW0,980
180 DRAW0,0
190 ENDPROC
200 DEF PROCteclado
210 REM JUGADOR UNO
220 IF INKEY(-51)=-1 THEN dx=4:dy=0
230 IF INKEY(-87)=-1 THEN dm=4:dn=0
240 IF INKEY(-66)=-1 THEN dx=-4:dy=0
250 IF INKEY(-70)=-1 THEN dm=-4:dn=0
260 IF INKEY(-67)=-1 THEN dx=0:dy=-4
270 IF INKEY(-34)=-1 THEN dx=0:dy=4
280 IF INKEY(-55)=-1 THEN dm=0:dn=4
290 IF INKEY(-102)=-1 THEN dm=0:dn=-4
300 x=x+dx:y=y+dy:m=m+dm:n=n+dn
310 ENDPROC
320 DEF PROCinicializar
330 x=100:y=490:m=1179:n=490
340 dx=4:dy=0:dm=-4:dn=0
350 col1=1:col2=2:borde=3
360 ENDPROC
370 DEF PROCleer—punto
380 pixel1=POINT(x,y)
390 pixel2=POINT(m,n)
400 IF pixel1 THEN PROCexplosion(x,y,1)
410 IF pixel2 THEN PROCexplosion(m,n,2)
420 ENDPROC
430 DEF PROCtrazar
440 GCOL0,col1:PLOT69,x,y
450 GCOL0,col2:PLOT69,m,n
460 ENDPROC
470 DEF PROCexplosion(ex,ey,cual)
480 FORI=1TO100
490 MOVEex,ey
500 GCOL0,RND(3)
510 PLOT1,RND(50)-25,RND(50)-25
520 NEXT
530 IF cual=2 THEN moto1=moto1+1
540 IF cual=1 THEN moto2=moto2+1
550 PROCinicializar
560 CLS
570 PROCtecla
580 PROCborde
590 PROCmarcador
600 ENDPROC
610 DEF PROCmarcador
620 PRINTTAB(1,0)"Moto Uno=";moto1
630 PRINTTAB(28,0)"Moto Dos=";moto2
640 ENDPROC
650 DEF PROCtecla
660 PRINTTAB(8,12)"PULSE CUALQUIER TECLA PARA EMP."
670 *FX21
680 A$=GET$
690 PRINTTAB(8,12)"
700 ENDPROC
```

Tortuga voladora

En este capítulo desarrollaremos un sencillo juego en el que la tortuga se pierde en el espacio

En nuestro juego *La tortuga del espacio*, ésta se encuentra perdida en la inmensidad del espacio, a mucha distancia de su base, a la que debe regresar. El juego exigirá que imprimamos varios mensajes en la pantalla.

No es de extrañar que la instrucción necesaria para este fin sea PRINT. Después de impreso el mensaje, se desplaza el cursor hasta el comienzo de la siguiente línea.

Para imprimir una sola palabra, a continuación de PRINT va la palabra propiamente dicha; por consiguiente, PRINT "HOLA imprime en la pantalla la palabra "HOLA". PRINT" se utiliza para imprimir la "palabra nula" (una "palabra" que no tiene caracteres).

El efecto de esta instrucción es, simplemente, imprimir una línea en blanco. Si se ha de imprimir más de una palabra, el texto se encierra entre corchetes para indicar que forma una *lista*:

```
PRINT [SU TIEMPO HA CONCLUIDO]
```

PRINT también se emplea para visualizar el contenido de una variable, de modo que PRINT :MARCADOR tomará el valor retenido en la variable "MARCADOR y lo visualizará. En la misma sentencia PRINT se pueden combinar mensajes y valores de variables encerrando toda la instrucción entre paréntesis, como en este caso:

```
(PRINT [SU MARCADOR FUE] :MARCADOR)
```

PRINT1 se comporta exactamente de la misma manera que PRINT, excepto que en este caso el cursor permanecerá al final del texto impreso y no se desplazará hasta la línea siguiente. Esto se puede demostrar entrando:

```
PRINT1 [COMO SE LLAMA UD?]
```

Operaciones de salida

Las instrucciones de LOGO, como HIDETURTLE o PRINT, hacen que suceda algo: se puede decir que las mismas tienen un efecto sobre la tortuga. Sin embargo, otras primitivas del LOGO (XCOR, p. ej.) no producen ningún efecto sobre la tortuga sino que producen un valor. Este valor se utiliza entonces normalmente como parámetro para una instrucción. Por lo tanto, digitar por ejemplo:

```
PRINT XCOR
```

haría que XCOR proporcionara el valor correspondiente a la coordenada actual *x* de la tortuga en la instrucción PRINT, que visualizaría entonces el resultado. Por consiguiente, si el valor corriente de XCOR fuera 20, PRINT XCOR haría que en la pantalla apareciera el número 20. Si se digitara XCOR sólo, aparecería el mensaje RESULT:20. Éste es en realidad un mensaje de error (las versiones LCSI imprimirán YOU DON'T SAY WHAT TO DO WITH 20: no dice qué es lo que hay que hacer con 20).

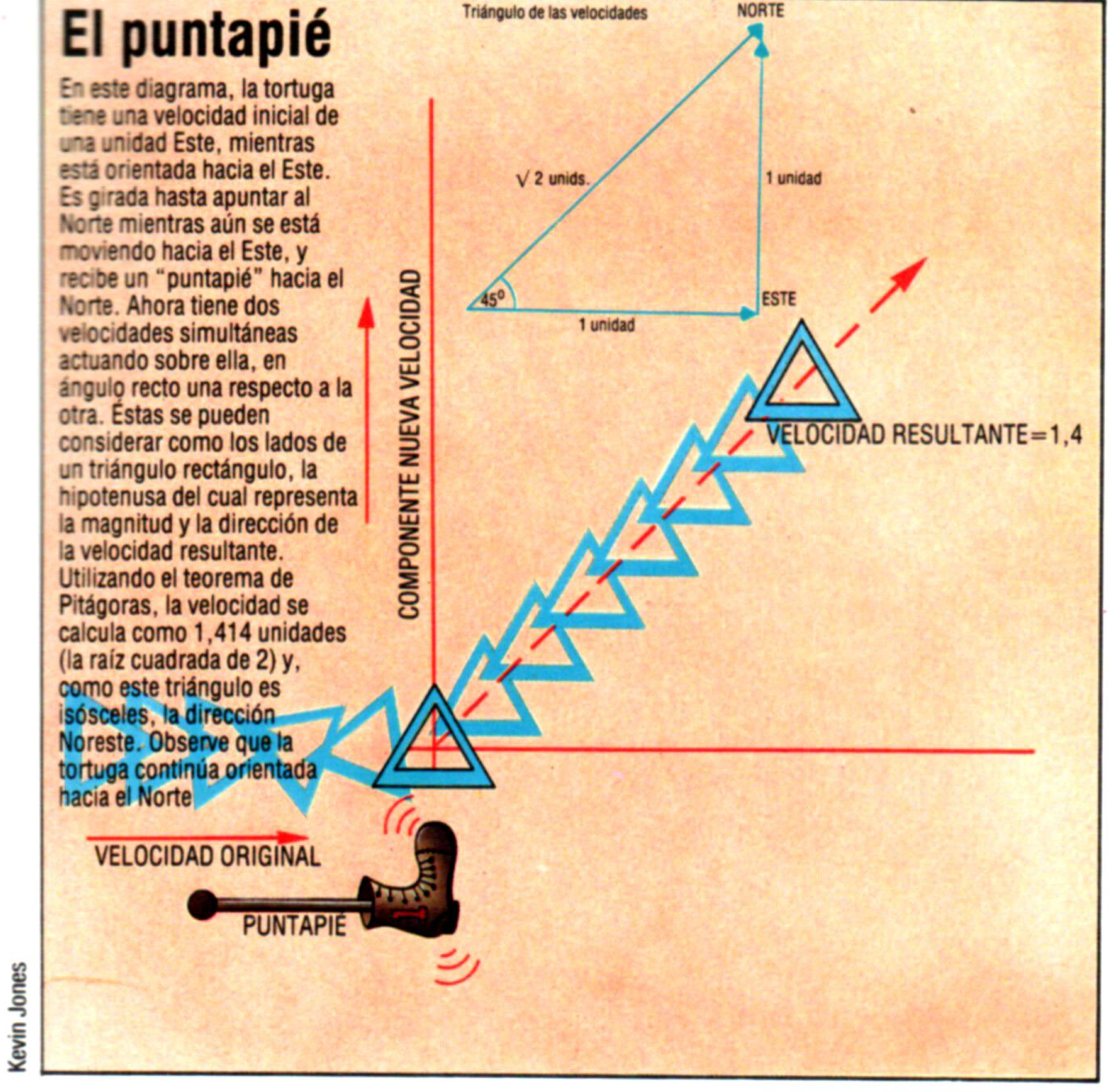
Todos los procedimientos que hemos escrito hasta ahora han sido instrucciones. Para crear *operaciones* debemos hacer uso de la primitiva OUTPUT. Como ejemplo sencillo, he aquí un procedimiento que proporciona la distancia desde el origen de la tortuga; este procedimiento utiliza SQRT para devolver la raíz cuadrada de un número:

```
TO DISTANCIA
  OUTPUT SQRT (XCOR*XCOR+YCOR*YCOR)
END
```

Intente desplazar la tortuga a distintas posiciones de la pantalla y utilice DISTANCIA para determinar a qué distancia se halla del origen. Por ejemplo, SETXY 30 40 PRINT DISTANCIA deberá dar como respuesta 50.

Cuando el LOGO ejecuta una instrucción OUTPUT, interrumpe la ejecución del procedimiento en curso, devolviendo el control al procedimiento que lo llamó. Esto se puede apreciar en el procedimiento MAX, que emite el mayor de dos números:

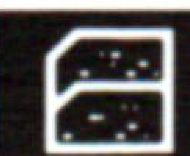
```
TO MAX :X :Y
  IF :X> :Y THEN OUTPUT :X
  OUTPUT :Y
END
```



El puntapié

En este diagrama, la tortuga tiene una velocidad inicial de una unidad Este, mientras está orientada hacia el Este. Es girada hasta apuntar al Norte mientras aún se está moviendo hacia el Este, y recibe un "puntapié" hacia el Norte. Ahora tiene dos velocidades simultáneas actuando sobre ella, en ángulo recto una respecto a la otra. Éstas se pueden considerar como los lados de un triángulo rectángulo, la hipotenusa del cual representa la magnitud y la dirección de la velocidad resultante. Utilizando el teorema de Pitágoras, la velocidad se calcula como 1,414 unidades (la raíz cuadrada de 2) y, como este triángulo es isósceles, la dirección Noreste. Observe que la tortuga continúa orientada hacia el Norte

Kevin Jones



PRINT MAX 6 2 dará 6 como resultado. Intente escribir un procedimiento para obtener el valor absoluto de un número, de modo que tanto PRINT ABS 4 como PRINT ABS (−4) devuelvan el valor 4.

Nuestro juego le pedirá que digite su nombre y pulse Return. He aquí un procedimiento:

```
TO TOMAR.NOMBRE
  SPLITSCREEN
  PRINT1 [COMO SE LLAMA UD?]
  MAKE "NOMBRE FIRST REQUEST
  (PRINT "HOLA :NOMBRE)
END
```

REQUEST espera a que se entre una línea, acabándola con un Return. Luego da salida a la línea en forma de lista. FIRST proporciona el primer elemento de una lista. A continuación pruebe el procedimiento TOMAR.NOMBRE y digite "Luisa" como nombre. Luego vea qué sucede si utiliza "Luisa Moreno" como entrada.

El juego controla el movimiento en pantalla de la tortuga mediante las teclas R, L y K. R hará girar la tortuga 30º en el sentido de las agujas del reloj (derecha); L la hará girar la misma cantidad en el sentido contrario (izquierda); K se utiliza para "darle un puntapié" a la tortuga, es decir, para aumentar su velocidad sea cual sea la dirección en la que esté apuntando. La tortuga se moverá alrededor de la pantalla y necesitaremos que responda de inmediato a estas teclas. Sería una ayuda si hubiera alguna primitiva del LOGO (READKEY —leer tecla—, quizá) que proporcionara la última tecla que se hubiera pulsado. Si fuera éste el caso, podríamos escribir:

```
TO INSTRUCCION
  MAKE "INSTR READKEY
  IF :INSTR="R THEN RIGHT 30
  IF :INSTR="L THEN LEFT 30
  IF :INSTR="K THEN PUNTAPIE
END
```

Por desgracia, ¡esta primitiva no existe! Pero podemos escribirla como un procedimiento:

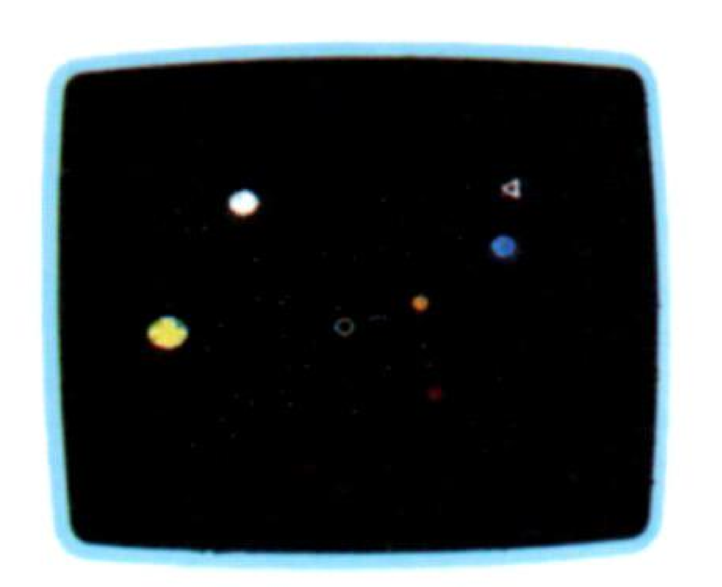
```
TO READKEY
  IF RC? THEN OUTPUT READCHARACTER
  OUTPUT"
END
```

Cuando se pulsa una tecla, ésta se almacena en el buffer del teclado. READCHARACTER simplemente saca el último carácter del buffer; si éste estuviera vacío, READCHARACTER esperaría a que se pulsara una tecla y después produciría el carácter correspondiente. RC? es cierto si el buffer contiene cualquier carácter, y es falso si está vacío. Así que READKEY producirá ahora el último carácter del buffer, o una palabra nula en caso de que éste esté vacío.

La tortuga espacial es en realidad una *tortuga dinámica*. Ésta es una tortuga que posee una velocidad, además de una posición y un encabezamiento como cualquier tortuga normal terrícola. La tortuga dinámica está en el espacio, de modo que no hay ni fricción ni gravedad. La tortuga dinámica obedecerá las leyes del movimiento de Newton.

Nuestra ilustración clarificará todo esto pero, a modo de ejemplo, vamos a suponer que la tortuga dinámica se está moviendo de izquierda a derecha a través de la pantalla a una velocidad de 1. Si se pulsa la tecla L, la tortuga gira para encararse hacia la parte superior de la pantalla, pero el impulso de la tortuga hará que se siga moviendo en su trayectoria horizontal. Entonces, si se pulsa K, la tortuga dinámica recibe un "puntapié" en la dirección en la cual está encarada. Esto resulta en un empuje hacia arriba de la pantalla de velocidad 1, y la tortuga dinámica se moverá diagonalmente a través de la pantalla a una velocidad de 1,4. La tortuga dinámica permitirá que usted experimente con un cuerpo que obedece las leyes de Newton; está diseñada para permitirle desarrollar una comprensión intuitiva de estas leyes sin que necesite comprender todas las ecuaciones correspondientes.

En el programa la velocidad de la tortuga dinámica se considera en términos de dos componentes a lo largo de los ejes x e y. Estos componentes se hallan utilizando las funciones SIN y COS. Los únicos controles del juego son los tres que ya hemos mencionado. Para empezar el juego, tan sólo digite START. Dispone de un tiempo fijo en el transcurso del cual debe alcanzar su objetivo, y el programa lleva el registro del mejor marcador conseguido.





Ian McKinnell

Tortuga extraterrestre
Tal como está impreso, el programa sólo generará la tortuga y el objetivo (su base). Las estrellas y los planetas que vemos en la fotografía se agregan utilizando algunos sencillos procedimientos de círculos

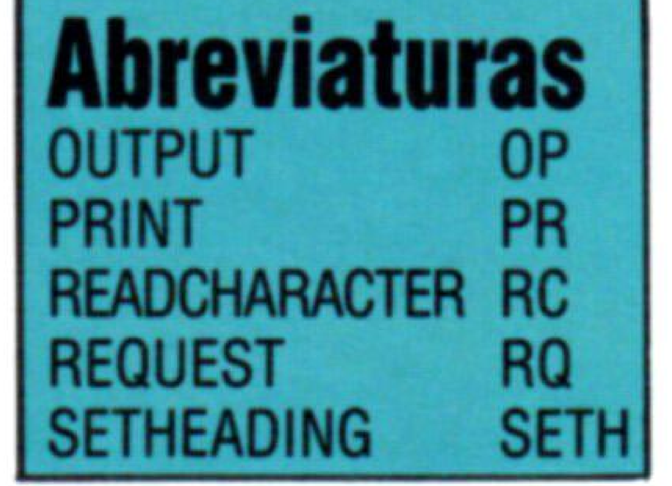
Complementos al LOGO

LOGO MIT	LOGO LCSI
DRAW	CS
PRINT1	TYPE
RC?	KEYP
READCHARACTER	RC
REQUEST	RL
SETHEADING	SETH
SETXY	SETPOS (seguido de lista)
FULLSCREEN	FS (inexistente en el Spectrum)
SPLITSCREEN	SS (inexistente en el Spectrum)

El LOGO LCSI, la sintaxis de IF es algo diferente; p. ej.:
IF DISTANCIA<5 [HAZ STOP]

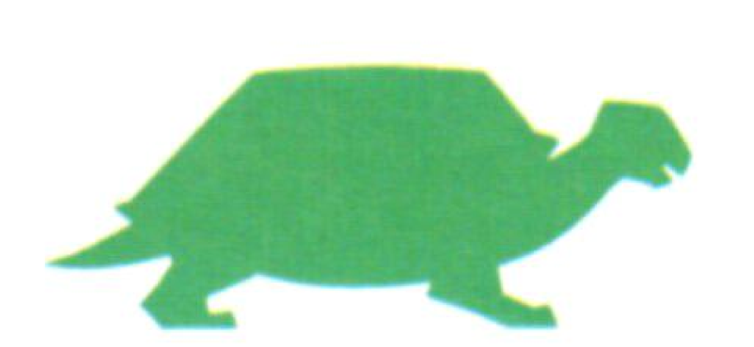
Proyecto de programa

Escriba un programa para jugar al *Lunar lander* (véase p. 832). He aquí un breve resumen:
El jugador está pilotando un cohete que se halla a cierta distancia por encima de la superficie de la Luna y que lleva una cantidad limitada de combustible. La gravedad ejerce un empuje constante sobre su nave e incrementa su velocidad de descenso en función de una cantidad fija por segundo. Pulsando F se encienden los motores del cohete. El objetivo del juego consiste en utilizar la tecla F de modo que el descenso sea lo suficientemente lento como para realizar un alunizaje seguro.



Abreviaturas

OUTPUT	OP
PRINT	PR
READCHARACTER	RC
REQUEST	RQ
SETHEADING	SETH



Respuestas a los ejercicios

1. Triángulos anidados

```
TO TRI :TAMAÑO :NIVEL
   IF :NIVEL=0 THEN REPEAT 3[FD :TAMAÑO
      RT 120] STOP
   TRI(:TAMAÑO/2)(:NIVEL-1)
   FD(:TAMAÑO/2)
   TRI(:TAMAÑO/2)(:NIVEL-1)
   RT 60
   TRI(:TAMAÑO/2)(:NIVEL-1)
   FD(:TAMAÑO/2)
   RT 60
   TRI(:TAMAÑO/2)(:NIVEL-1)
   LT 60
   BK(:TAMAÑO/2)
   LT 60
   BK(:TAMAÑO/2)
END
```

2. Copo de nieve cuadrado

```
TO NIEVE 1 :TAMAÑO :NIVEL
   REPEAT 4[LADO1 :TAMAÑO :NIVEL RT 90]
END
TO LADO1 :TAMAÑO :NIVEL
   IF :NIVEL=0 THEN FD :TAMAÑO STOP
   LADO1(:TAMAÑO/3)(:NIVEL-1)
   LT 90
   LADO1(:TAMAÑO/3)(:NIVEL-1)
   RT 90
   LADO1(:TAMAÑO/3)(:NIVEL-1)
   RT 90
   LADO1(:TAMAÑO/3)(:NIVEL-1)
   LT 90
   LADO1(:TAMAÑO/3)(:NIVEL-1)
END
```

3. Curva sin gradiente en ningún punto

```
TO W :XPASO :YPASO :NIVEL
   WARRIBA :XPASO :YPASO :NIVEL
   WABAJO :XPASO :YPASO :NIVEL
END

TO WARRIBA :XPASO :YPASO :NIVEL
   IF :NIVEL=0 THEN SETXY (XCOR+:XPASO)
      (YCOR+:YPASO)STOP
   WARRIBA(:XPASO/6)(:YPASO/2)(:NIVEL-1)
   WABAJO(:XPASO/6)(:YPASO/2)(:NIVEL-1)
   WARRIBA(:XPASO/6)(:YPASO/2)(:NIVEL-1)
   WARRIBA(:XPASO/6)(:YPASO/2)(:NIVEL-1)
   WABAJO(:XPASO/6)(:YPASO/2)(:NIVEL-1)
   WARRIBA(:XPASO/6)(:YPASO/2)(:NIVEL-1)
END
TO WABAJO :XPASO :YPASO :NIVEL
   IF :NIVEL=0 THEN SETXY(XCOR+:XPASO)
      (YCOR-:YPASO)STOP
   WABAJO(:XPASO/6)(:Y PASO/2)(:NIVEL-1)
   WARRIBA(:XPASO/6)(:YPASO/2)(:NIVEL-1)
   WABAJO(:XPASO/6)(:YPASO/2)(:NIVEL-1)
   WABAJO(:XPASO/6)(:YPASO/2)(:NIVEL-1)
   WARRIBA(:XPASO/6)(:YPASO/2)(:NIVEL-1)
   WABAJO(:XPASO/6)(:YPASO/2)(:NIVEL-1)
END
```

La tortuga del espacio

```
TO COMENZAR
   MAKE“MAX 0
   MAKE“MEJOR”
   DRAW
   HT
   OBJETIVO
   JUGAR
END

TO OBJETIVO
   PU SETXY 0 5 PD
   RT 90
   REPEAT 36 [FD 31.4/36 RT 10]
   PU
END

TO JUGAR
   TOMAR.NOMBRE
   INIC
   CONDUCIR
END

TO TOMAR.NOMBRE
   SPLITSCREEN
   PRINT1 [COMO SE LLAMA UD?]
   MAKE“NOMBRE FIRST REQUEST
END

TO INIC
   MAKE“MARCADOR 200
   SETXY 100 100
   SETH 270
   MAKE“XVEL 0
   MAKE“YVEL 0
   FULLSCREEN
   ST
END

TO CONDUCIR
   ORDEN
   MOVIMIENTO.TORT.DIN
   IF DIST<5 THEN HAZ STOP
   MAKE“MARC:MARC-1
   IF :MARC=0 THEN T AGOTADO
   STOP
   CONDUCIR
END

TO ORDEN
   MAKE“ORD READKEY
   IF :ORD=“R THEN RIGHT 30
   IF :ORD=“L THEN LEFT 30
   IF :ORD=“K THEN PUNTAPIE
END

TO READKEY
   IF RC? THEN OUTPUT
   READCHARACTER
   OUTPUT”
END

TO PUNTAPIE
   MAKE“XVEL+3*SIN HEADING
   MAKE“YVEL+3*COS HEADING
END

TO MOVIMIENTO.TORT.DIN
   SETXY XCOR+:XVEL
   YCOR+:YVEL
END

TO DISTANCIA
   OUTPUT SORT
   (XCOR*XCOR+YCOR*YCOR)
END

TO HAZ
   PRINT”
   SPLITSCREEN
   (PRINT[BIEN HECHO]:NOMBRE)
   (PRINT[SU MARCADOR FUE]
   :MARCADOR)
   INFORME
   OTRAVEZ
END

TO INFORME
   IF :MARC>:MAX THEN MAKE
   “MAX:MARC MAKE”
   MEJOR:NOMBRE PRINT”
   (PRINT[MARC MAS ALTO ES]
   :MEJOR[CON]:MAX[PUNTOS]
END

TO OTRAVEZ
   PRINT1[OTRA PARTIDA?]
   MAKE“RESP FIRST REQUEST
   IF :RESP=“SI THEN REPETIR
      STOP
   IF :RESP=“NO THEN STOP
   PRINT [DECIDETE, SI O NO?]
   OTRAVEZ
END

TO TIEMPO.AGOTADO
   PRINT”
   SPLITSCREEN
   PRINT[SU T HA CONCLUIDO]
   OTRAVEZ
END

TO REPETIR
   HT
   TOMAR.NOMBRE
   INIC
   CONDUCIR
END
```



Subir hasta cero

Dejando, por el momento, el análisis del direccionamiento, estudiaremos con más detalle cómo funcionan las pilas

Hasta ahora nuestro uso de los dos registros índices de pila, el S y el U, se ha limitado a servirnos de ellos como registros índices adicionales. Sólo muy de pasada hemos mencionado el empleo de la llamada *pila hardware* como almacén de direcciones de retorno tras las llamadas a subrutinas. Se impone volver sobre nuestros pasos y estudiar la arquitectura de una pila, así como el modo de utilizarla.

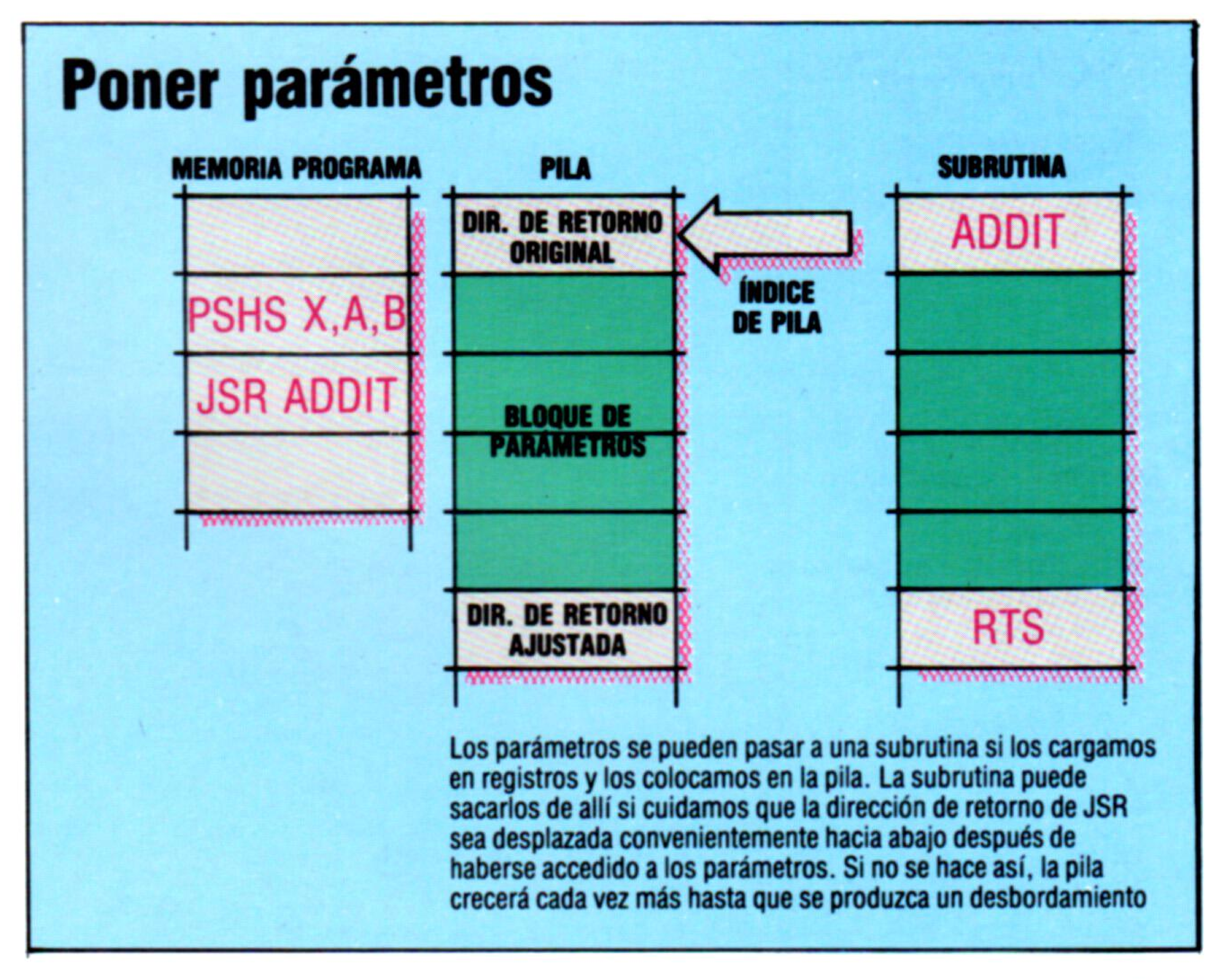
Una pila es un tipo específico de estructuración de datos dentro de la categoría de las *listas*. Aun cuando nada sepa de lenguajes cada vez más populares que se basan en el tratamiento de listas (p. ej., el LISP y el LOGO), usted tiene sin duda una idea común de lista, puesto que simplemente se trata de una secuencia de ítems de datos. Tal secuencia puede ordenarse según alguna propiedad común a los datos (así, p. ej., de menor a mayor si se trata de datos numéricos; por orden alfabético, si son formados con caracteres alfanuméricos), o bien puede ordenarse aleatoriamente según vayan integrándose a la lista. En todas las listas es claro que podemos distinguir entre "siguiente" y "anterior", y en particular entre el primer elemento de la lista y el último (llamados "cabeza" —*head*— y "cola" —*tail*—).

Una característica importante de una lista es que es una *estructuración dinámica de datos*; es decir, los ítems de datos pueden añadirse o eliminarse de ella arbitrariamente. En una lista general los datos se colocan o se sacan de cualquier posición que ocupen en ella. Lo que convierte una lista en *pila (stack)* es que los datos sólo pueden ser introducidos o extraídos por uno solo de sus extremos. Cada nuevo ítem añadido a la pila se convierte en "cabeza de lista", y sólo éste puede sacarse de ella.

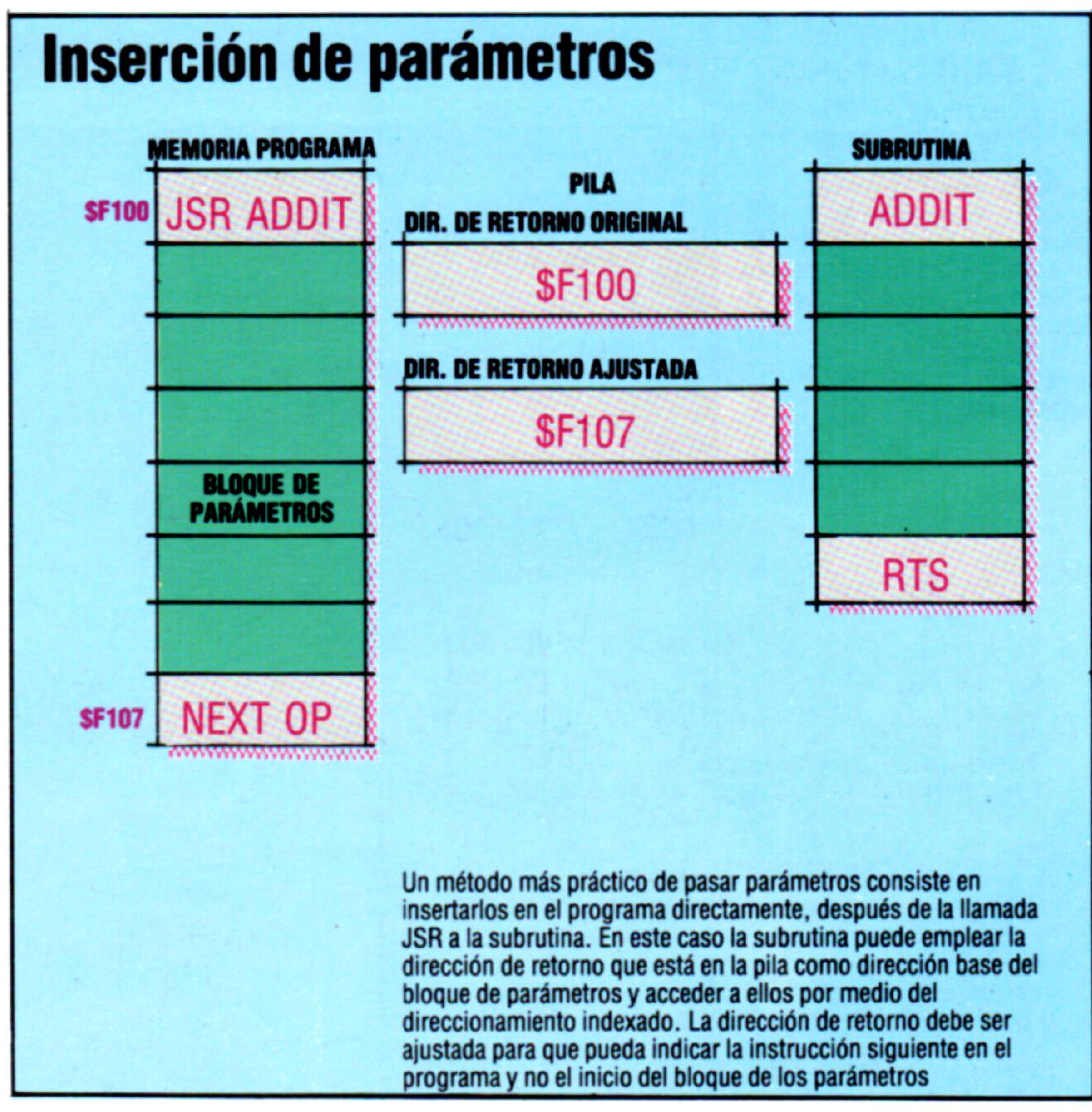
Su mismo nombre nos da una idea del modo como funciona. Considérese una pila de platos en un cajón: según se van necesitando los platos éstos se van tomando por arriba, y encima sólo se ponen platos limpios. Desde luego que usted podría extraer un plato de la parte intermedia de la pila, pero esto sería absurdamente dificultoso. Lo que sí es posible es inspeccionar un determinado dato en cualquier posición de la pila.

En el funcionamiento de la pila pueden presentarse dos situaciones extremas: el caso en que se vacía la pila, donde no hay problema alguno si se añade un nuevo dato en la siguiente operación de pila, pero si ello no es así, puede resultar problemático; el otro caso extremo es que la pila se llene a rebosar. Esta segunda hipótesis es fácil de imaginar volviendo a nuestra pila de platos: habrá un momento en que ésta alcance el techo y sea imposible añadirle ningún plato más.

Las pilas de los ordenadores funcionan de modo análogo. Las operaciones de poner o de sacar datos

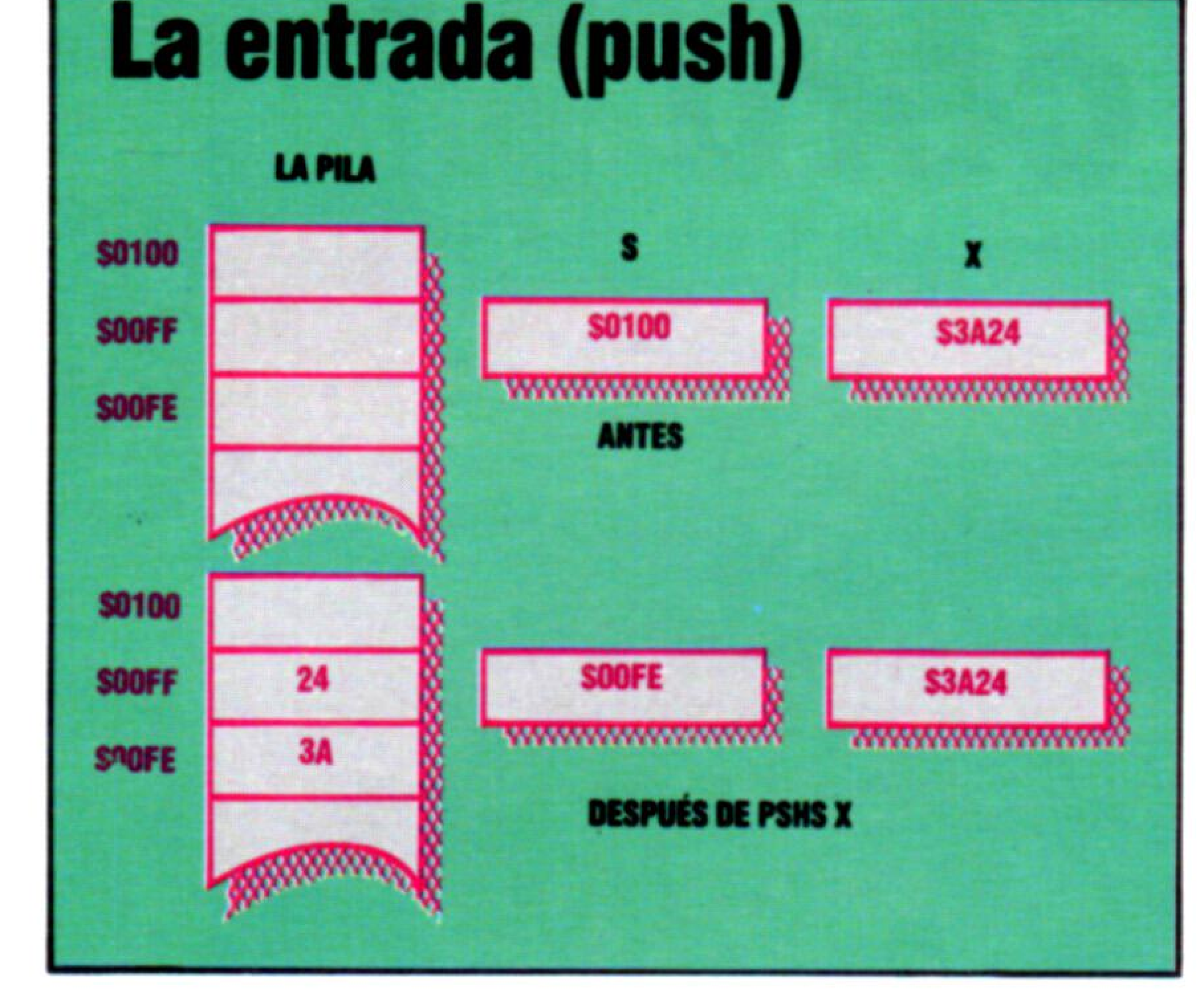


Los parámetros se pueden pasar a una subrutina si los cargamos en registros y los colocamos en la pila. La subrutina puede sacarlos de allí si cuidamos que la dirección de retorno de JSR sea desplazada convenientemente hacia abajo después de haberse accedido a los parámetros. Si no se hace así, la pila crecerá cada vez más hasta que se produzca un desbordamiento

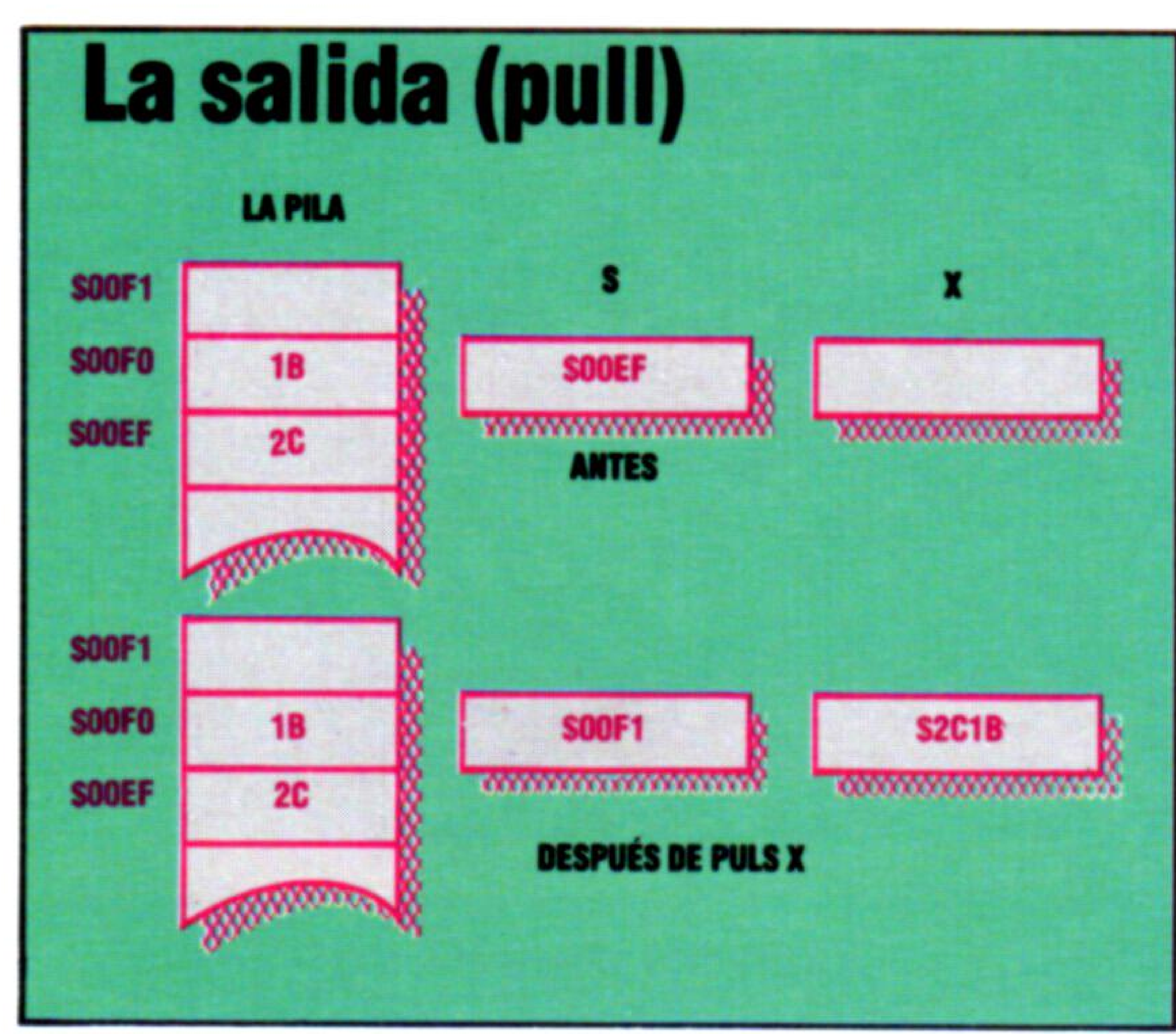


Un método más práctico de pasar parámetros consiste en insertarlos en el programa directamente, después de la llamada JSR a la subrutina. En este caso la subrutina puede emplear la dirección de retorno que está en la pila como dirección base del bloque de parámetros y acceder a ellos por medio del direccionamiento indexado. La dirección de retorno debe ser ajustada para que pueda indicar la instrucción siguiente en el programa y no el inicio del bloque de los parámetros

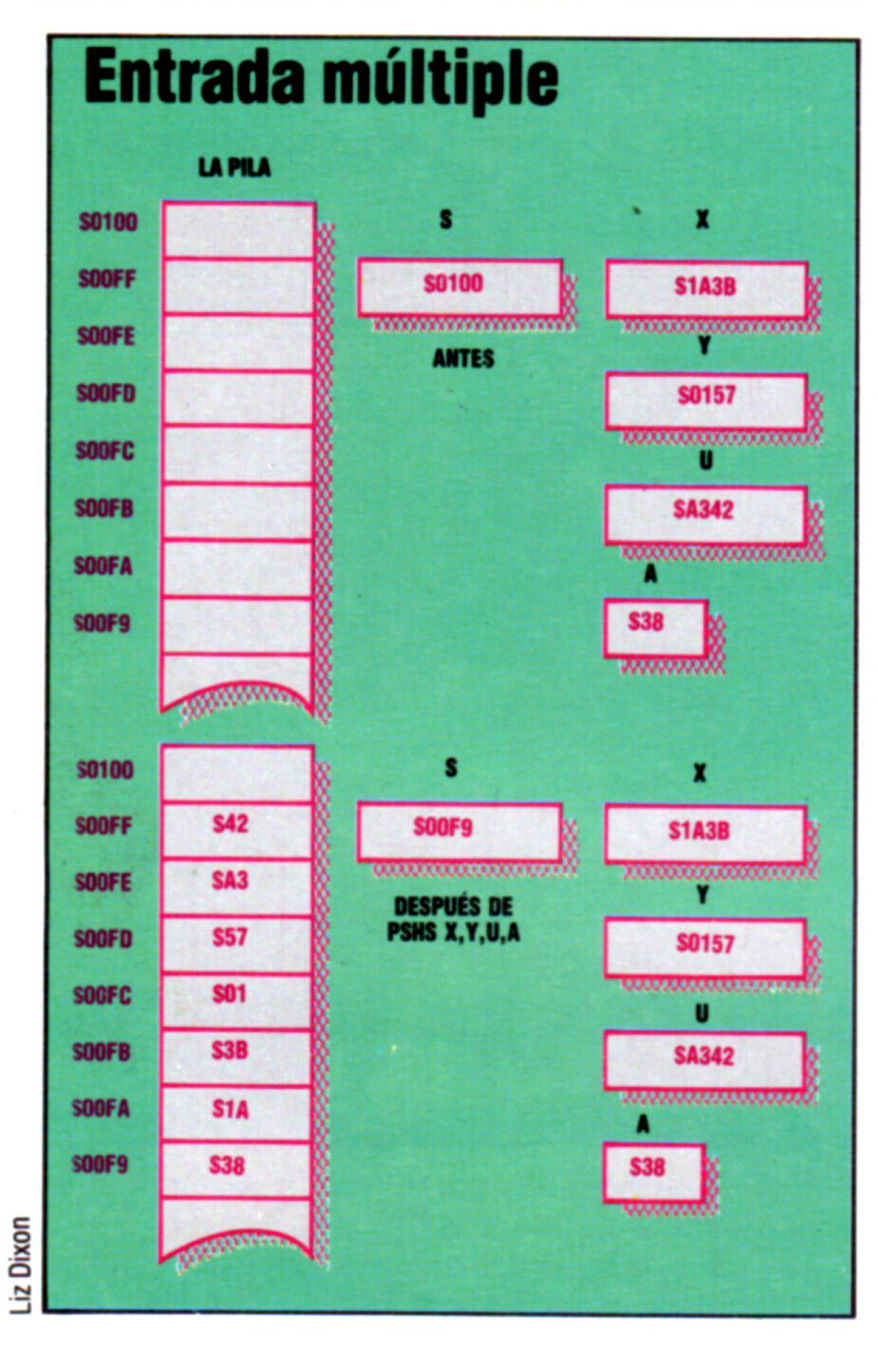
Empujón va
El índice de pila del 6809 siempre apunta a la "cima" (*top*) de la pila, es decir, al último byte que entró. Una vez ejecutada PSHS X, entonces S se decrementa en dos unidades, para poder indicar la nueva cima, mientras que el contenido de X (que es un registro de dos bytes) es escrito en dicha dirección según el formato *hi-lo*. Luego, efectivamente, la pila "sube hasta cero", pues el indicador de pila señala las posiciones inferiores de la memoria según va creciendo la pila



Empujón viene
Una vez ejecutado PULS X, el contenido de los dos bytes que están en la dirección del índice de pila es copiado en X, mientras que S es incrementada en dos unidades para indicar la nueva cima de la pila



En tropel
Cuando se ejecuta una operación de pila con múltiples registros, los registros en cuestión son accedidos según un orden predeterminado: PC,U o bien S,Y,X,DP,B,A,CC. Si es ejecutada PSHS X,Y,U,A, los primeros en entrar en la pila serán los contenidos de U seguidos de Y,X y A



de ellas se conocen como operación de "poner" y de "sacar" (*pushing* y *pulling/popping*) respectivamente. Y las situaciones extremas mencionadas se conocen como "desbordamiento" (negativo o normal) de datos (*underflow* y *overflow*).

La implementación de las pilas puede realizarse de varias maneras (p. ej., empleando una matriz de una dimensión en un programa BASIC), pero el método que estamos estudiando exige un bloque de memoria disponible y un registro que podemos designar como *índice de pila*. Este apuntador se hace imprescindible para conocer la posición que ocupa el dato en cabeza. A diferencia de la pila de platos, la de una memoria no puede ser inspeccionada porque no existe ningún indicio que permita distinguir una posición que contenga un determinado dato de la siguiente a ella, la cual hasta puede que no forme parte de la pila. Vale la pena señalar que, al igual que en una operación de "carga" de un registro desde memoria, donde en realidad el dato no es "extraído" de la memoria, sino copiado, así también los datos en realidad no son "sacados" de la pila, sino que se cambia el índice de la parte superior de ésta.

Por tanto, el índice de pila contiene la dirección de la actual cima (*top*) de la pila. Dos son las posibles variaciones: éste puede facilitar la dirección de la siguiente posición libre para almacenar datos o bien la dirección del último ítem de datos almacenados en la pila. Esta última indicación es la que se ha escogido por convenio para el 6809, sabiendo que no existen ventajas al preferir éste en lugar del otro método: otros procesadores emplean la técnica alternativa.

Entre una pila de platos y la de memoria existe una diferencia significativa en cuanto a organización, y es que en esta última y dentro del sistema del 6809 el crecimiento es hacia abajo: cuantos más datos van poniéndose a la pila, más bajas son las direcciones señaladas por el índice. Es lo que se entiende por "crecer hasta cero".

Operaciones de la pila

Las dos operaciones de una pila del 6809 se representan por PSH (*push*: poner) y PUL (*pull*: sacar). Ambas pueden emplearse con cualquiera de los dos índices, el S y el U, por lo que distinguiremos entre PSHS, PULS, PSHU y PULU. Los datos sobre los que se opera deben proceder de (o ir a) un registro, aunque es posible poner o sacar de varios registros a la vez mediante una sola instrucción.

La instrucción PSHS X primero hará que S se decremente, o sea el índice de pila se decrementa en dos unidades (o en una, si el registro colocado en la pila es de ocho bits) para apuntar a la dirección de la siguiente posición libre de pila; y además hará que el contenido de X quede almacenado en dicha dirección. El primer diagrama ilustra esta operación. De nuevo es de notar el convenio del direccionamiento *hi-lo* para el 6809: el byte superior, o *hi*, de X (o sea, $3A) queda almacenado en $00FE, es decir, una posición más abajo que el byte inferior, o *lo*, que es $24 y queda almacenado en $00FF. Si se sirve de un ensamblador, estos detalles de si los indicadores de pila incrementan o decrementan carecen de interés: el ensamblador se encarga de preparar convenientemente la memoria.

La instrucción PULS X tiene el efecto contrario: el



valor de 16 bits contenido en ese momento en S es cargado en X, incrementando seguidamente S en dos unidades. Es lo que ilustra el segundo diagrama.

Se puede poner (o sacar) más de un registro a la vez. Por ejemplo en la siguiente instrucción:

```
PSHS X,Y,U,A
```

En casos como el presente en que más de un registro es colocado en la pila, se desprecia el orden en que son especificados los registros y en su lugar se observa este otro orden: PC (registro *c*ontador del *p*rograma), U o bien S,Y,X,DP (registro de página directa), B,A y CC (registro de *c*ódigo de *c*ondición). Naturalmente son sacados por este mismo orden pero a la inversa. La única limitación consiste en que ni S ni U pueden ser colocados en la pila.

Las pilas se emplean en la programación general por ser lugares muy convenientes para el almacenaje rápido temporal, pero el principal servicio que prestan se revela al tratar con interrupciones (de las que hablaremos más adelante) y subrutinas. Ya hemos visto cómo es colocado en la pila el contenido del registro contador del programa en el momento de llamar una subrutina, y cómo se extraía de ella cuando se abandonaba ésta (RTS equivale a PULS PC). Cualquiera de las dos pilas puede servir para pasar parámetros a la subrutina, pero en especial S.

Al método hasta ahora seguido para pasar parámetros a través de los registros (como, p. ej., el programa de la Tabla de Salto de p. 1119) suelen hacerse dos reparos importantes. El primero es que puede haber más parámetros que registros, y el segundo es que puede resultar engorroso el hecho de que la rutina llamada emplee un registro donde se encuentre un parámetro que es necesario conservar. Hay, sin embargo, otras dos técnicas conocidas para pasar parámetros:

1) Los datos pueden almacenarse junto al código del programa, justo después de la llamada a la rutina, mediante el uso de las directivas FCB, FDB o FCC. El valor del registro contador del programa, que es llevado a la pila por medio de la instrucción JSR, facilita la dirección del primero de estos valores (ya que PC siempre señala el byte siguiente al de la instrucción en curso) y con desplazamientos apropiados sirve para obtener las restantes direcciones. El primer programa que damos como ejemplo ilustra esta técnica. Habrá de cuidarse que se organice de tal modo la instrucción RTS que pase el control a una instrucción real y no a un grupo de datos.

2) Los datos pueden ser cargados en registros y ser llevados a la pila antes de la llamada de la subrutina, y de ella se sacarán para ser utilizados en la subrutina. Aquí la atención se pondrá en que con la instrucción RTS el índice de pila accederá a la dirección de retorno del PC previamente guardado en la pila. Esta operación se ilustra en el segundo ejemplo. Se trata de un método mucho más útil que el primero.

En ambos procedimientos, el doble papel desempeñado por S y U, como registros índices y como índices de pila, significa que los datos llevados a la pila pueden ser referenciados por medio del direccionamiento indexado además de poder accederse fácilmente a ellos para ser sacados de la pila. Lo cual permite asegurar que se dejaron en la pila los ítems correctos para el momento del regreso.

Suma de precisión múltiple

He aquí dos fragmentos de lenguaje máquina para ilustrar dos métodos alternativos de realizar una suma de precisión múltiple por medio de pilas. En el primer fragmento, los parámetros son colocados en ellas después de ser llamada la subrutina. Una llamada para sumar dos números de cuatro bytes en $100 y $104, dejando el resultado en $108, sería:

```
        BSR    MPADD
        FCB    4               ; Long. de cada núm. (bytes)
        FDB    $100            ; Dir. del primer número
        FDB    $104            ; Dir. del segundo número
        FDB    $108            ; Dir. del resultado
        ORG    $1000
MPADD   PSHS   X,Y,U,A,B,CC    ; Guarda todos los registros (introduce 9 bytes en la pila)
        LDU    9,S             ; Cargará en U la dir. de retorno de PC colocada en la pila por la instrucción BSR
        PULU   X,Y,B           ; Se tratan datos que siguen a la llamada de subrutina como si fueran una pila, de modo que se obtienen las dirs. de los dos núms. y se introducen en X e Y, así como la long. en B
        LDU    ,U              ; Dir. del resultado queda en U
        ANDCC  #%11111110      ; Limpia el flag de arrastre
BUCLE   LDA    ,X+
        ADCA   ,Y+             ; Realiza suma de un byte más otro, con posibilidad arrastre
        STA    ,U+             ; Almacena el resultado
        DECB                   ; Compr. si suma ha concluido
        BNE    BUCL            ; Si no lo está, BUCLE
        LDU    9,S             ; De otra manera toma dir. de retorno y añade 7 para pasar por encima de los 7 bytes de parám. almac. tras la llamada
        LEAU   7,U
        STU    9,S
        PULS   PC,U,X,Y,A,B,CC ; Restaura todos los registros al estado que tenían antes, incluido PC en vez de usar un RTS de más
```

Este segundo ejemplo realiza la misma operación pero colocando los parámetros en la pila. La secuencia de llamada sería:

```
        LDU    #$108           ; Dirección del resultado
        LDX    #$100           ; Dirección del operando
        LDY    #$104
        LDA    #4              ; Longitud
        PSHS   U,X,Y,A         ; Coloca los parámetros en la pila
        BSR    MPADD
        ORG    $1000
MPADD   PSHS   X,Y,U,A,B,CC    ; Guarda todos los registros
        LEAU   11,U            ; Guardados los registros, toma 9 bytes, y la dirección de retorno toma 2 bytes, en total 11 bytes. Así la actual U apunta a los parámetros
        PULU   X,Y,B           ; Procede como anteriormente
        ANDCC  #%11111110
BUCLE   LDA    ,X+
        ADCA   ,Y+
        STA    ,U+
        DECB
        BNE    BUCLE
        PULS   PC,U,X,Y,A,B,CC
```

La guerra nuclear ha comenzado

Este terrorífico juego transcurre durante una conflagración nuclear y consiste en impedir la destrucción de la humanidad

El éxito de un juego diseñado para salas recreativas a menudo depende de lo acertada que sea su modalidad de "atracción" (la visualización que aparece en la pantalla mientras no se está jugando). El juego debe, asimismo, producir una adicción instantánea. Si fracasa en uno solo de estos aspectos será desechado enseguida por el propietario de la sala en favor de una máquina más rentable. A pesar de que ya ha pasado el momento en que *Comando de misiles (Missile command)* alcanzara su máximo éxito en las salas recreativas, durante un tiempo fue muy popular. La versión para las máquinas Atari es testimonio de su antigua gloria.

El escenario del juego es sencillo y sutil a la vez. El jugador se sitúa en la posición del comandante de una estación antimisiles durante una guerra nuclear y debe proteger a seis ciudades de la amenaza de destrucción haciendo explotar misiles antinucleares que salen al paso de las cabezas explosivas que se acercan. El diseño pretende satisfacer tanto la megalomanía como el ansia de aventuras latentes en un aficionado a los juegos de "marcianitos", se encuentre éste en una sala recreativa o en su casa.

Durante el juego, la pantalla muestra una vista de sección transversal de la acción, que ilustra las seis ciudades y una estructura piramidal en el centro con los misiles antibalísticos listos para su lanzamiento. El rastro de los misiles que se acercan aparece luego desde la parte superior de la pantalla. El jugador desplaza una cruz alrededor de ésta valiéndose de la palanca de mando. La cruz se coloca en el recorrido de los misiles que se aproximan y, accionando el pulsador de disparo, se lanza un misil antibalístico desde la base de lanzamiento del jugador. El proyectil estalla en las coordenadas de la cruz, destruyendo todos los misiles que se acerquen y estén dentro del radio de alcance de la explosión.

Sin embargo, algunos misiles enemigos están provistos de cabezas explosivas múltiples que se dividen en varias trayectorias, cada una de las cuales puede destruir una ciudad entera. El juego se complica aún más con la aparición de satélites "asesinos" y bombarderos enemigos que vuelan a poca altura, capaces de lanzar nuevas olas de proyectiles nucleares. Se otorgan puntos por la destrucción de aviones y misiles enemigos. Al final de un ataque, un marcador muestra la cantidad de ciudades que uno ha conseguido proteger con éxito de la aniquilación y cuántos misiles antibalísticos le quedan.

A medida que el jugador va avanzando a través de los seis niveles de dificultad que posee el juego, los misiles atacantes se desplazan con mayor rapidez y va aumentando la cantidad de cabezas explosivas en las que se dividen. En este punto es necesario desarrollar una estrategia general, en vez de destruir cada misil por separado. El jugador puede optar, por ejemplo, por disponer una "barrera" de misiles antibalísticos que exploten en línea y, con suerte, neutralicen la oleada en una sola acción contraofensiva.

Además, en los niveles superiores el juego se complica por la aparición de cabezas explosivas lanzadas con paracaídas. Éstas son sumamente difíciles de destruir: si su misil estalla ligeramente fuera del objetivo, es probable que sea "soplado" fuera de ruta (presumiblemente debido a la corriente de aire ascendente) y, por lo tanto, es necesario hacer explotar un misil justo encima de uno de éstos.

En el transcurso del juego debe recordar que sólo dispone de un número limitado de misiles antibalísticos (30 en el primer nivel) y si éstos se acabaran estará condenado a presenciar cómo el enemigo aniquila la base de misiles y las ciudades.

Cada nivel se compone de dos olas de ataque separadas, después de las cuales se calcula el marcador. Al igual que en otros juegos Atari, se puede "saltar" a un nivel de juego superior. Cada nivel se distingue por sus diferentes colores de fondo y primer plano. El juego termina cuando se destruyen las seis ciudades, y esta conclusión inequívocamente pesimista se refuerza con la pantalla final, que visualiza una apocalíptica explosión y las palabras *THE END*.

El juego se vende en cartucho para los ordenadores Atari y viene con un manual de instrucciones en color claramente superior a la documentación que acompaña a la mayoría de los otros juegos.

Comando de misiles: Para los ordenadores Atari
Editado y distribuido por: Atari, AUDELEC, Compás de la Victoria, 3, 29012-Málaga, España
Autores: Atari
Palanca de mando: Necesaria
Formato: Cartucho

Liz Dixon

mi computer

Editorial Delta, S.A.



10057
9 788485 822836